An Introduction to Artificial Neural Networks for Image Processing

Keith W. Cunningham, PhD
kwc@mobile-map.com

Abstract

Artificial Neural Networks (ANN) are a type of computer programming that is designed to mimic the process of human cognition found the natural neural network – the human brain. Artificial Neural Networks have been successfully applied to tasks involving image compression, classification, pattern matching, and change detection. This presentation summarizes the current research of the author related neural networks for image processing. These interests include extracting feature primitives from raster images and LiDAR, the chunking of data into larger groups of information, and ultimately change detection.

Introduction

Artificial neural networks represent a very different form of computer programming. Neural networks are an unstructured approach to programming unlike the structured programming environments of C++, Visual Basic, *.NET, etc. The unstructured approach makes them very interesting tools for performing a variety of complex logic operations, especially those operations that may not be reducible to a series of rules or steps that can be sequentially coded in a serial, logical method.

Neural networks are good for solving unstructured problems, making quick approximations, and storing data. They are suited for pattern recognition in both signals and pictures. They excel with issues of change detection and data optimization problems like the traveling salesman and the Chinese postman. A variety of other applications exist for neural networks that offers a wide range of opportunities for remote sensing and geographic information systems.

A special ability of the neural network includes their ability to learn. By teaching a neural network what outputs are expected from many and varied inputs, their programmer is actually their teacher.

Structure & Terminology

Neural networks are designed to emulate the human cognitive process. As a computation model, neural networks consist of many interconnected simple processors, similar to the neurons in the human brain. These neurons are capable of performing threshold logic operations like yes or no. Input/output functions can be performed via the internal arrangement of the connections between the neurons, which are termed "synapses."

Neural networks are often described as operating in parallel or in a continuous fashion, because they are capable of accepting many parallel inputs, processing the inputs in parallel, and outputting continuous results in near real-time. However, most neural networks are implemented on serial computers (Intel, Motorola, etc.), so they only simulate parallel processing.

Inputs for a neural network are taken at the input layer, processed by neurons in the hidden layer, and results are directed to the neurons in the output layer. The structure of the connections between neurons in the hidden layer determines input-output associations. Different internal structures generate different responses. A network has to be trained to process inputs and determine outputs by the synaptic "wiring" among the neurons.

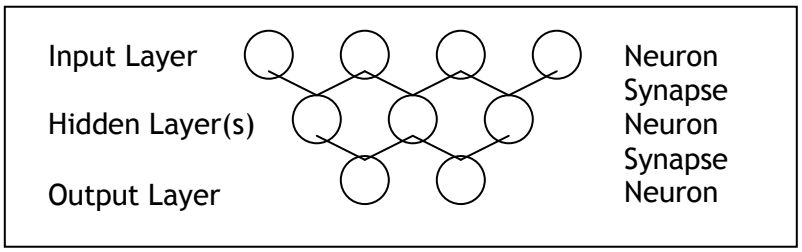

Figure 1 – Basic Architecture

There are two basic types of neural networks: the feedback and the feed-forward. Both networks are typically organized into "layers" of neurons so that neurons in one layer are connected to neurons in another layer. The feed-forward network moves inputs though the synapses and neurons in only a forward direction. The feedback network moves inputs and outputs in both directions through the network. Other types of network structures, such as the "fully-recursive" have more specialized functions, such as data compression.

An important characteristic distinguishing the feedback from the feed-forward network is how each learns. Generally, the feedback neural network can learn more quickly because errors can be back propagated through the network, changing the synaptic wiring according learning based on errors. Feed-forward neural nets reorganize their connections randomly, and training continues until the desired output is achieved.

Learning

The process of wiring the neurons to generate specific responses is called training. Before training, the connections between neurons are randomly set, thus these connections generate outputs that are not associated with the inputs. During learning in networks, the inputs are fed to an untrained network, and the random connections are modified so that specific outputs become associated with specific inputs. By rearranging the network's synaptic connections and their connection strengths, the network can learn and forget. Some networks are designed to continuously learn while others are created to forget some information if that information is not recalled frequently.

Since the neurons are simple binary processors, the mechanism used to influence the decision-making process of the neuron is based on "weights" of the synapses connecting different neurons. The biological and electrical analog of the synaptic weights is based on the fact that thicker electric wires (nerves) with more insulation carry signals more efficiently than thinner wires without insulation.

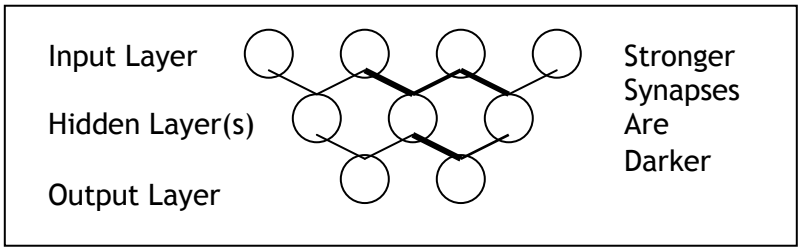

Figure 2 – Synapse Strengths

The efficiency of signal transmission in the synapse is reflected by numeric weights and mathematical functions. The geometry of these mathematical functions determines how signals are conveyed and is termed the "transfer function." Transfer functions can be linear, step-like, curved, and sigmoid. The transfer function is important because it describes how a signal is modified, including intensification and diminution of neuronal connectivity. Transfer functions work differently for different applications and selecting the proper type of transfer function may sometimes be determined only via trial and error. A nuance with this process is the fact that different transfer functions may associate different layers and neurons in the network.

Upon successful training, an internal equilibrium is achieved. This equilibrium represents the mathematical weights of each synapse connecting each neuron, based upon the inputs and outputs. This equilibrium represents the knowledge of the network and is termed an "associative memory" or "content-addressable memory" because the information describing the associations between input and output is incorporated into the structure of the processors that perform the input/output function. In effect, the connective matrix is an energy surface that defines how an input will be guided across this surface to a specific output.

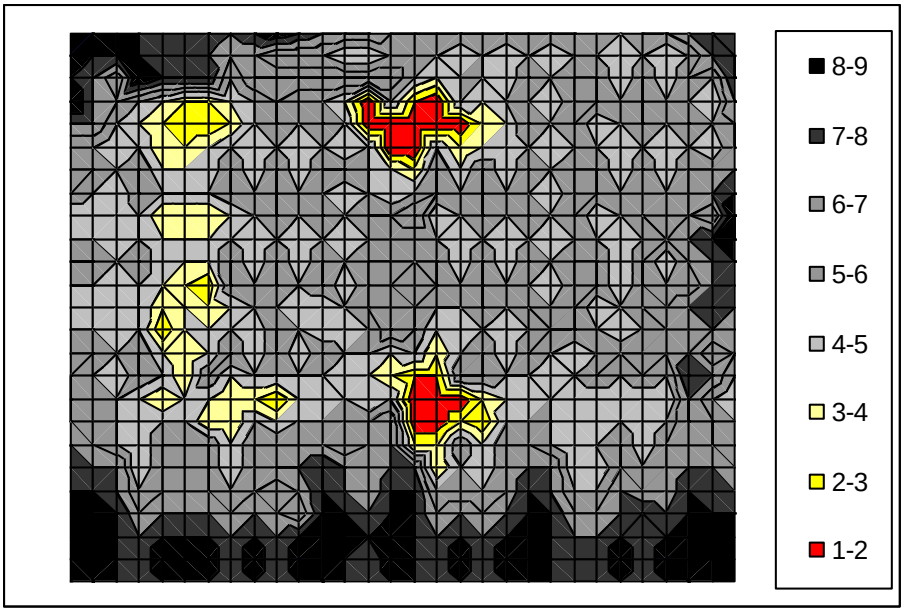

Figure 3 – Energy Surface – A Pachinko Game

Types of Learning

Two basic types of learning characterize neural networks. Unsupervised learning requires networks to learn on their own without output guidance from the operator. Supervised learning requires the operator to provide output responses for each input.

Unsupervised Learning - Both feed-forward and feedback neural networks are specialized in determining the most efficient organization of input data on their own, without training supervision by a human. In these unsupervised networks, data are loaded into the network and network processes the inputs until the network independently reaches an equilibrium, representing the most compact mapping of the data. These self-educating – genetic – networks are best for data compression and change detection. The unsupervised learning process is analogous to a child learning a spoken language.

Supervised Learning - The other approach for training is a process supervised by a human operator. Inputs are fed to an untrained network, which has randomly determined connection strengths between neurons. The operator determines a desired training response for each input and if the output from the network does not match the desired output, the internal network topology is adjusted. The supervised learning process is similar to how a child memorizes the multiplication tables.

During training, the inputs are moved though the network in a forward direction and upon reaching the output neurons the response is compared to a training response. In the case of feedback networks, the error is propagated backwards through the network and the connections among the neurons are rewired according to the error. The feed-forward network also rewires itself, but without the benefit of knowing the extent of error encountered, as the feedback network does. The input and rewiring processes are continuously repeated until the network creates a stable internal structure that generates the desired output with each input – and the most efficient mapping of the data.

Supervised learning is the most common type AAN training. It is faster than unsupervised training and is implemented more efficiently in the backwards-propagation networks than in feed-forward networks. This occurs because people

who are building neural networks to perform specific input-output functions want immediate results, and supervised learning with a "back-prop" network can "rewire" its synaptic matrix more quickly and accurately than the feed-forward networks.

In some types of networks, there may not be designated input and output neurons. Such networks are termed "recursive," meaning every neuron is connected to every other neuron. Such "fully-recursive" networks learn to classify inputs by simply processing the data until the system reaches "equilibrium" or its energy state is at the lowest-possible level. Occasionally, a network becomes hung-up at some local energy minima that does not reflect the ultimate equilibrium that the network is capable of achieving. In this case, the user must vigorously test the network to determine how well it has learned and adjust the training data and learning parameters (i.e., transfer functions) until a lower energy minima is reached.

Sample Applications

Neural networks have a variety of applications. Because they behave in a parallel, continuous fashion, neural networks are better for solving some types of problems than serial programs using symbolic, serial logic. The following figure describes the functional differences between neural networks and conventional serial programs.

Characteristic	Parallel Computing	Serial Computing
Processor	Many (infinite) – Simple	Few (singular) – Complex
Steps	Few iterations	Many (thousands)
Failure	Graceful & beautiful	Catastrophic
Training	Dynamic by Example	Explicit with Serial Logic
Solutions	Multiple & Simultaneous	Singular & Sequential
Structure	Self-organized & Dynamic	Formalized & Rigid

Figure 4 – Serial vs Parallel Computing

Neural networks have been used for commercial applications since the late 1950s. One of the first practical neural networks was ADALINE, which was designed to remove noise from telephone lines. Today they are used in optical character recognition systems, handwriting recognition, financial pattern analysis, and by airlines to maximize profits in seat reservations. Neural networks designed to detect change are being used in industrial quality assurance programs to inspect images and listen to sounds in situations where a human would become bored or inattentive after inspecting products for long periods of time. In the financial industry, neural networks to predict patterns are being used to predict when stocks should be bought and sold.

A spectacular example of the "content-addressable memory" of neural networks is an experiment with the Bible. The entire text of the Bible was compressed by a neural network into a structure only 1.5 megabytes in size. A user can provide only a portion of any passage from the Bible and the neural network can then retrieve the entire passage of scripture instantaneously (McCord-Nelson 1991). Another example of the efficiency of neural networks is the traveling salesman solution that has a statistical combinatorial solution. Yet the neural network can process the inputs until it reaches an internal equilibrium and determine optimal solutions because the mapping of the salesman's destinations and paths of travel in the neural network represents the most efficient mapping of the problem.

Mapping Applications

Neural networks have a variety of applications for image processing, feature extraction and change detection. As neural networks become more robust and easily created, other mapping applications such as change detection, temporal analysis, and a variety of predictive applications will emerge.

The more common mapping applications include feature detection and classification. This process requires the training of a neural network with examples of the objects to be mapped. This approach has been used successfully in classifying pixels in images collected by satellites, ortho-imagery, and even for the "cloud-of-points" problem with laser scanners and LiDAR.

The simplest approach for image processing with neural networks is to perform analysis based on the signal patterns for each pixel in an image. For instance, a neural network is trained to look for specific spectral signatures among the different bands of data describing each pixel and when a specific set of spectrums match what the network is looking for, the pixel is classified. This approach has been used to calculate the aerial extent of impervious surfaces in urban areas for storm-water run-off studies, determine if fields have been cultivated, and to classify regions of vegetation based on texture.

A more complex approach to pixel classification observes the characteristics of groups of pixels simultaneously for classification. This approach may only classify a central pixel or a tight convolution, but the classification process uses inputs from the neighbors, or surrounding convolution. Thus the surrounding pixels can "weight" or influence the classification of "point of interest." A similar technique using statistical methods such as bi-linear interpolation and cubic convolution is often used to enhance the edges of features in raster imagery. This approach with neural networks can be used to perform more detailed and specific classifications. This 'con'-textural approach is appropriate in removing noise from scanned documents because examples of pixel noise can be presented to neural networks for training and during operation. Those pixels identified as noise can be modified or have their values reduced to zero – especially in the case of bi-tonal imagery.

An even more interesting use of neural networks in map conversion is scene classification. In this example, an entire array of pixels can be presented simultaneously to the neural network. Depending on the features found in the scene, it could be classified and further processed to yield individual pixel classifications. So a 100-pixel array can contain enough information for the extraction of feature primitives, such as lines and edges. This is the approach taken by the author with manipulating data from laser scanners, including LiDAR.

Change detection in images of different vintages is another process that has been automated with neural networks. When a map image is stored in a neural network, that image is precisely represented by the architecture of the network – and an equilibrium map is created. Any changes in a more recent image is noted against the original, thus detected by the neural network. Such experiments have been performed to note changes in scanned images of currency, so that any change in a $100 bill, even at the level of a single pixel, could be detected and the difference noted. The Bible passage network described earlier is an example of this type of network: the structure of the network defines the data. By simply supplying a unique part of the pattern you are searching for, the network can complete the pattern, determine differences in the pattern, or tell you that no such pattern exists – immediate recall.

Data Preprocessing

A drawback of neural networks is the amount of data preprocessing required before a network can be trained and operated. This is especially true for neural networks designed to process imagery. Preprocessing data for a neural network involves the scrubbing, filtering, aggregation, accentuation, and transformation of the data. Preprocessing can accentuate or conceal data as well as massage data in such a way as to improve the performance of the network.

Data preprocessing is a subject rarely discussed in the literature on neural networks. Commercial neural network applications do describe what types of data can be used by the network, but getting one's own data into a "flavor" which the network "likes" can be complex and arduous. Typically it seems that each user's data and needs are distinct enough to require a unique approach to the preprocessing of their data. The reason data preprocessing is particularly important for image processing by neural networks is the fact that a variety of data structures exist for digital images and these

must be transformed into strings of values for processing. Completing this transformation without suffering loss of data integrity or spatial relationships requires the special handling of the imagery.

Preprocessing an image for use by a neural network may require several steps. First, the source must be scanned, as with ortho production. Unfortunately the process of scanning the photo can actually eliminate important data as the imagery is compressed from a native 24-bit format. Then a variety of statistical filters and convolutions may be applied to the image to further sharpen features, remove noise, and re-map colors to another palette. Even more complex mathematical functions have been applied to images in preparation for neural network processing, such as edge-detection filters that employ techniques to calculate the zero-crossings of second derivative functions. Ultimately, the image must be transformed into a file format that can be read and understood by the neural network and this is yet another transformation step.

The Math – Statistical Mechanics

The basic approach when utilizing a neural networks for image processing is a statistical processes. Multivariant analysis, Gaussian classifiers, bi-linear interpolations, and convolutions were among the original methods used to classify patterns and features. Stochastics and transformations work well when the features to be classified can be clearly distinguished from other features by their spectral signatures. However serial data processing approaches can be slower and less robust compared to neural networks.

Because the neural network is a simulated parallel processor (some hardware implementations of neural networks do fully operate in parallel), it has definite advantages. The number of instructions –lines of code – determines processing speed with a serial algorithm created where each instruction takes a clock cycle to complete. In a trained neural network that has only five layers of neurons, only five clock cycles are required to process all of the inputs, perhaps 10s, 100s, or even 1000s. Thus a neural network operates in a real-time environment compared to the serial program.

Underlying the function of the neural network is a mathematical matrix defining the relations between all of the neurons via the synaptic transfer functions. This mathematical matrix is an array of connectivity probabilities – a branch of mathematics termed "statistical mechanics." Once such a matrix of probabilities is created, it can readily be incorporated into more formalize, symbolic, serial programming techniques. Thus the best of both programming and classification environments can be achieved – the formalized world utilizing off-the-shelf Active-X controls in a C++ environment with the beauty of a cognitive, learning model with few processing steps.

Summary

As a tool, neural networks are still gestating. Though neural networks appear to have a variety of uses such as signal processing and determining the most efficient organization of data, they constitute only one of many tools available for the processing of imagery and producing derivative features from the "clouds of points" generated from laser scanners. By no means have they supplanted the use of structured programming, but the future of neural networks appears richer and stronger because of their ability to process data in parallel, learn, and be embeddable in traditional programming environments.

As my research into neural networks continues, their limitations and possibilities continue to be revealed. Applications integrating neural network classifiers, filters, and feature detectors may become common in remote sensing and GIS applications. Neural networks in the future may be a tool of choice for creating a variety of databases extracted directly from imagery, reducing or eliminating the need for human interpretation.

