A Processor Approach to build an Artificial Neural Network

Alexandre Mano

Departamento de Informática, Universidade do Minho
4710-057 Braga, Portugal
mi6076@di.uminho.pt

Abstract. Artificial Neural Networks (ANNs) have followed two radically dissimilar paths: software simulation and dedicated hardware. This communication introduces the main concepts behind the ANN idea and analyses one processor specifically built as a hardware tool for creating a Neural Network, capable of replicating the human's brain operations. It also tries to assess its current possibilities within the industrial framework and possible future applications.

1 Introduction

Learning in biological systems involves adjustments to the connections between neuronal connections in the brain. That's true for Artificial Neural Networks (ANNs) too. Learning in an ANN occurs by example through training, achieved by exposure to a set of input-output data.

In general, ANNs are good pattern recognizers and reliable classifiers, with the ability to generalize in making decisions about less-than-precise input data. They offer reasonable solutions to a multitude of classification problems such as speech, character and signal recognition, as well as functional prediction and system modelling where the physical processes are not yet understood or are highly complex. Neural networks can also be applied to control problems, where the input variables are measurements used to drive an output actuator, and the network learns the control function. The advantage in ANNs lies in their robustness against distortions in the input data and their leaning capability. They can behave quite well at solving problems that are too complex for conventional technologies (problems that don't have an algorithmic solution or for which the algorithmic solution is too complex to be found) and are often suitable to problems that people are good at solving, but for which traditional computational methods are not.

ANNs can be implemented in software or in specialized hardware. This communication explains the concepts behind the ANN technology and deals with a dedicated multilayer perceptron ANN, implemented within a PCI board, built to act as a coprocessor in a microprocessor system, as a low cost, but efficient solution to the problems depicted above.

2 What is an Artificial Neural Network?

An Artificial Neural Network (ANN) is a mathematical abstraction of the interconnection between neurons found on human's brains. It tries to replicate the way the biological neurons work by creating computerized counterparts, with some specific functions:

- Neurons are connected to each other by an unidirectional connection, much like the biological neurons, that are linked by their axons;
- Neurons have one or more inputs and one output only;

- Each connection from one neuron to another has an specific associated weight, that can be changed during the ANN life-cycle;
- The introduction of an additional fixed input (bias) allows the transfer function to be the same to all neurons.
- Each neuron computes its possible output (or 'activation') as the weighted sum of their inputs;
- When the activation value exceeds a given limit, the neuron 'fires' and produces an output whose value is mapped by a transfer function.

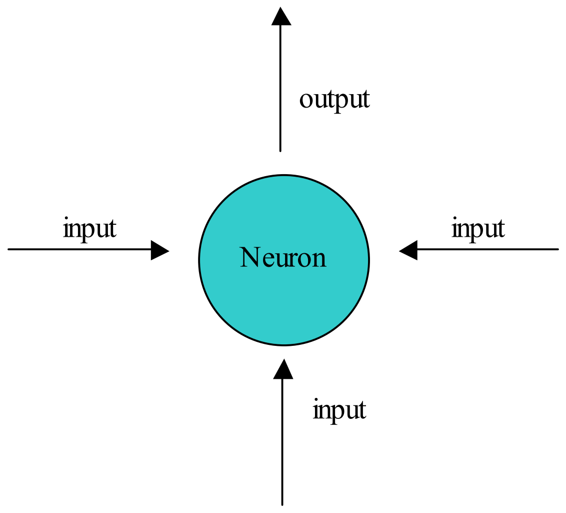


Fig.1. Representation of a 3-input artificial neuron.

There are various different types of ANNs. Some ANNs are classified as feedforward, while other are recurrent (i.e., implement feedback) depending on how the data is processed through the network. Another way of classifying ANN types is by their method of learning (or training), as some ANNs employ supervised training while others are unsupervised or self-organizing. The ANN discussed in this communication is a recurrent, supervised one, based around the multilayer perceptron (MLP) paradigm.

The MLP ANN typically comprises many layers, which are divided into:

- one input layer (layer 0), that receives the external impulses or signals;
- one or more hidden layers, that act as the ANNs computing and learning nodes;
- one output layer, that transmits the computed result to the exterior.

This multi-layered approach gives the ANN the following properties:

- each neuron presents a smooth, nonlinear output;
- the hidden layers help the network learn complex problems by progressively identifying meaningful aspects from the input data set;
- the large number of connections between neurons determines a high degree of interconnectivity; a change in a connection influences the entire population of connections and their weights.

2.1 The learning process and the minimization of errors

A multi-layered ANN can solve any computable problem through a supervised learning process. The learning process is accomplished by feeding the ANN with inputs and evaluating the resulting outputs, balancing its weights in the process and assuring that this balance provides the network the capability of giving reasonable responses to outputs that it has not seen before. This is done in a series of steps that try to minimize the difference between the desired output values and the actual values computed by the ANN. This task can be achieved using techniques such as exhaustive search, back propagation and search methods.

Exhaustive search explores all possible values of the weights and then evaluating which set of values yields the minimum error. While it would theoretically find the best set of weights in every possible situation, its usage is too expensive in computational terms.

Back propagation (BP) consists of two phases. One forward step which applies an input pattern to the network through its input layer, resulting in a set of values at the output layer that are compared to the intended target values. The backward step corrects the weights considering the output values and the expected ones. The initial assignment of weights is a random one.

If the desired output vector (D) and the actual vector computed by the ANN (A) are considered, a common choice for a function (e) that computes the error between them is

$$e(D,A) = \sqrt{D^2 - A^2} \ .$$

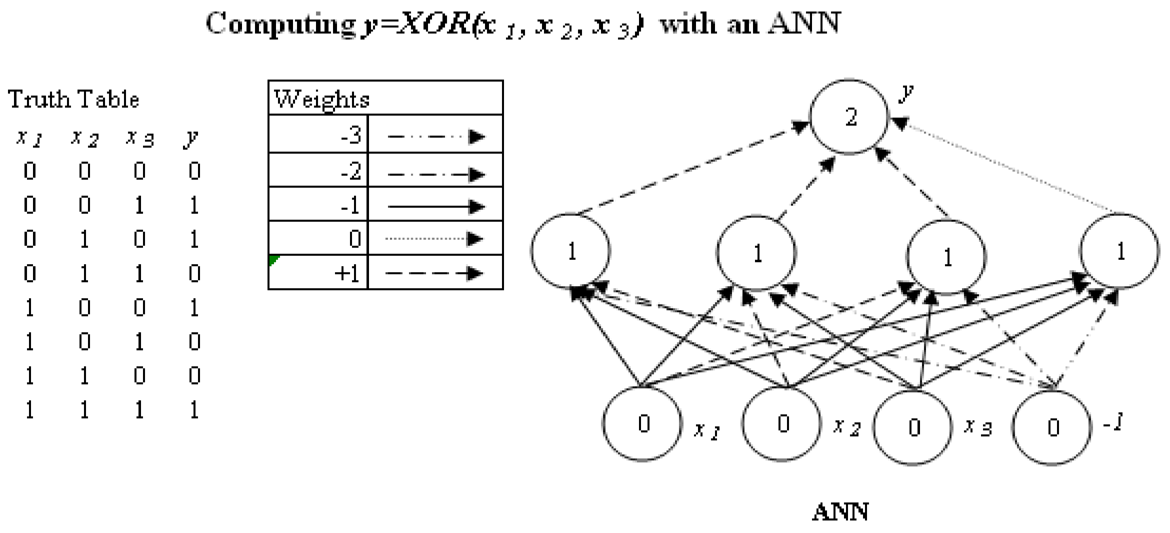
By changing the weights, the ANN attempts to bring *e* as close to zero as possible. This process of weight adjustment can be extended to the more common situation where there is a set of input vectors (I_1, I_2, ..., I_n) and a set of desired output vectors (D_1, D_2, ..., D_n). If the set of actual output vectors (A_1, A_2, ..., A_n) is taken into account, the error (E) is commonly chosen to be the average for the various errors (e) calculated by the equation above. This learning process takes place repeatedly until the weights and activation values stabilize and the error of the output values converges to some pre-determined minimum value. This technique is frequently used, and is also known as steepest gradient method. Its main problem is that balancing the weights so that each iteration of the algorithm brings the error closer to zero is a difficult task.

Finally, the *search-based methods* start with a random weight assignment and compute the error function for some neighbours of that point, i.e., sets of output values closest to the one computed. The best neighbour is chosen for the next iteration and the process is repeated until no further progress is possible. This technique is simpler than BP, but it requires a very fast evaluation of the set of input-output vectors for each neighbour.

These last two techniques have another disadvantage. The algorithms used can get stuck in a situation known as ‘local minimum’. This happens when the search for lower values of the error function discussed above lead to an infinite loop involving a number of sets of input-output values. If the search is caught in such a loop a deadlock situation arises that halts any progress.

To avoid this problem, methods like the Reactive Tabu Search (RTS) have been developed. RTS combines the search for the lowest possible value of the error and the storage of the moves made while searching. The reverses of the moves are prohibited for a fixed number of iterations (the *tabu list size*) to avoid cycles. The adaptable reaction approach increases the list size when local minima cause repetitions in all areas of the search space. It can also lead the search through a region of the search area where the error actually increases, if this takes the search out of limited areas where it became trapped.

The example below shows an ANN after the weight balancing has been done. It computes $y = XOR(x_1, x_2, x_3)$ and the transfer function used is $f(a) = \begin{cases} 0 \text{ if } a < 0 \\ 1 \text{ if } a \geq 0 \end{cases}$.



Truth Table

x_1	x_2	x_3	y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Fig. 2. Example of an ANN implementing a simple logical function

In some situations, this learning process alone is not enough to produce a good generalization, i.e., to ensure that error at the output is within the minimum required by the specifications of the ANN. When this happens, changes in the topology of the ANN, in the learning algorithm, or both, must be considered.

2.2 Designing the ANN

Before the ANN can be used to solve any problem, some aspects have to be decided upon. Some of the most important are the topology of the network, the transfer function chosen for the neurons and the size of the input-output training set.

When deciding upon the number of nodes an ANN should have, some compromise must be reached. More nodes in each layer allow more accurate associations between inputs and outputs and allow the network to learn the training examples very well. Generally, it can be said that the more nodes the network has, the better its memorization abilities are. On the other hand, more nodes require more communication and computational time and sometimes it can cause the network to perform poorly when new inputs are presented. This happens because the more nodes it has, the more connections and, therefore, the more weights need to be computed and balanced, and the ANN can lose generalization capabilities.

When choosing the transfer function, it typically has one or more parameters that can be tuned. However, given the ANNs usual adaptability, the tuning of the transfer function is not a major problem. The overall computation of an ANN is fairly tolerant to changes in the transfer function.

In general, more vectors for training the ANN accounts for better results. It is crucial to the system's performance that the training vectors are statistically representative of the entire possible input universe. The training methods used for the ANN must be robust enough to accommodate for some erroneous or wrongly-defined data in the training set.

3 A Hardware Implementation of an ANN

NeuriCam has developed a processor that performs all of the ANN tasks described above: it is the NC3001 processor.

The NC3001 processor is a coprocessor that can interact with the PC's main processor, acting as a neural processor optimised for neural computation. Each NC3001 has several processors (artificial neurons) operating concurrently and connected in a way that mimics the neuronal operations inside the human brain.

The processor depends on two factors for performance:

- A highly-parallel VLSI computing engine with simple fixed-point processing units and on-chip memory for storing weights.
- Usage of the Reactive Tabu Search (RTS) algorithm combined with standard training techniques.

The NC3001's main technical features are:

- Pipelined digital data stream, Single Instruction Multiple Data (SIMD) architecture.
- 32 fixed-point fully-parallel multiply-and-accumulate processors (MACs) operating in parallel from a common broadcast bus.
- 32 Kbit on-chip dynamic RAM memory for weight storage closely coupled with the processors. Memory can be assigned to a single neuron or be partitioned among several neurons to implement multi-layers network with a single chip.
- 16 bit data, 8 bit weight and 16 or 32 bit results.
- 1200 million multiply-and-accumulate operations per second with a 40 MHz clock.
- Capability of operating as a coprocessor in microprocessor systems.
- Support circuitry for external look-up table (LUT) RAM to implement the transfer function of neural network.
- 70 mm2 for chip's area and 250.000 transistors.

3.1 Chip description

Each processing element (PE) which is used to implement the neurons on the NC3001 contains a Multiply and Accumulate (MAC) unit and the weight memory. They work as an unit to produce an output result that allows implementation of multi-layer networks, like the model discussed in the first part of this communication. The processing is carried through by a fully-parallel multiplier architecture, which provides a speed of one operation per clock cycle. The time required for a full multiply-accumulate is N_{inp} clock cycles, where N_{inp} is the number of inputs. The MACs operate on two's complement signed integers. The basic multiplier cell employs fully static 28-transistor CMOS adders. For high-speed operation, one level of pipelining has been included in the multiplier array. The 32-bit accumulator uses the same adder used in the multiplier array but with two additional levels of pipelining.

Dynamic RAM is used for the weights to provide a reasonably fast capability, which is crucial during the training phase of the network. In the recognition mode of operation, non-volatile memory would be sufficient. The memory depth required for one network layer is equal to the maximum number of neuron inputs which are implemented. In the NC3001, one dedicated on-chip memory is associated to each neuron. The weight memory was partitioned into 2-Kbit blocks closely coupled to their processors. The output of the RAM is pipelined to hide the access time of the memory.

The global bandwidth required to feed the weights to the MACs during training is N_{neuron} input-output operations per accumulation cycle, which can cause a serious I/O bottleneck when external memory is used with such highly parallel architectures. This problem has been dealt with in the NC3001 by using localized, internal memory for the weights. This enables a high bandwidth between the MACs and memory at the expense of increased chip area. The calculation of the changes in the weights during the learning phase is driven to other circuits in the system, such as the host processor.

A 32-to-16 bit barrel shifter is inserted in the output channel to scale the results and convert them back to a word length, suitable for further processing in multi-layer networks.

The transfer function is implemented in an off-chip, memory-based look-up table (LUT). Memory paging can be used to store a number of different transfer functions. The architecture's minimum configuration is achieved combining a chip and the LUT. The modular approach (one processor-one memory) used allows more powerful architectures to be built combining a number of modules in parallel. The increase in overall system performance is a linear function of the number of processors used.

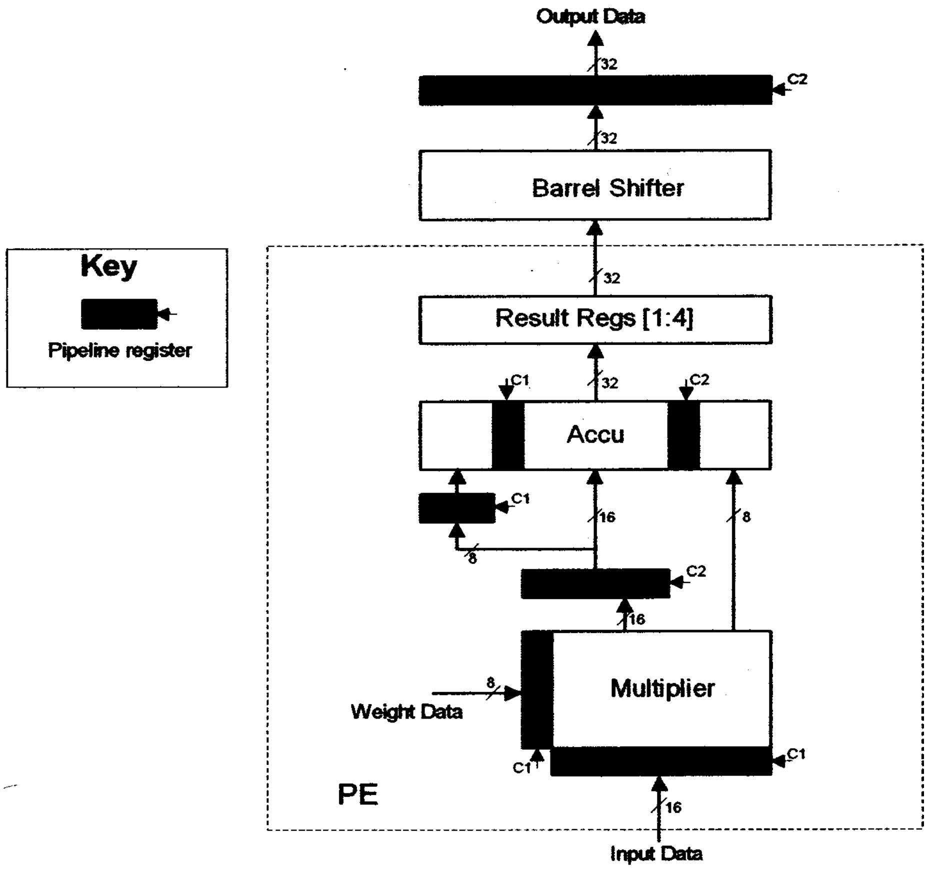


Fig.3. NC3001 pipeline structure. (Image courtesy of NeuriCam)

This architecture is integrated on the NC3001 as an array of 32 parallel processing units with associated weight memory and control logic. The word widths have been optimised for learning with RTS: the 16-bit width of the broadcast bus is adequate to represent signals from transducers and intermediate results between layers. The 8-bit memory word is sufficient for many classification tasks and the 32-bit word width of the output channel permits accumulation capacity.

As described above, the NC3001 acts as a coprocessor. Its current version is shipped in a dedicated PCI board, which has two chips, each with the full 32-processor/neuron architecture built in. The board itself communicates with the motherboard via a 32-bit interface.

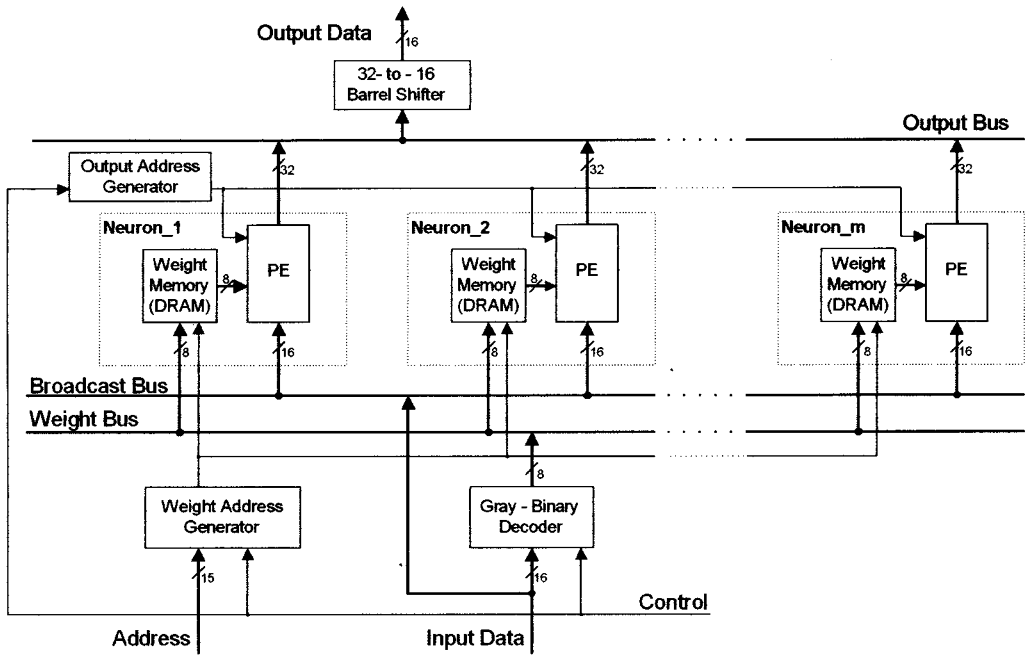


Fig. 4. Block diagram of the NC3001. (Image courtesy of NeuriCam)

3.2 Applications

Software emulation is often too slow to be of any real practical value in many applications involving fuzzy computation and optimisation. A dedicated processor like the NC3001 speeds up the critical portion of the calculation while maintaining a low system cost.

Present users operate on speech recognition terminals; on-line industrial quality control of flat products; control systems for laser spectroscopy; triggering and event classification in physics; data filtering and enhancement in scientific instrumentation for meteorology; finger tracking for optical pointing devices; licence plate recognition systems; ECG signal compression and transmission; time series analysis for stock market forecasting.

Some of the most promising markets for ANNs in general and for usage of the NC3001 in particular are digital image processing, analysis and compression; speech recognition in noisy environments for the automotive industry; fingerprint and face recognition and user validation systems; railway track control systems; access control systems for public transport stations; compression of domain-dependent biomedical signals; signal processing and conditioning for scientific instrumentation.

Other potential markets are support systems for the disabled; optical character recognition; signal processing for home videoconferencing, telemedicine, intelligent vehicle control, pollution and environmental control.

3.3 Future developments

The planned innovations of the new products are based on better fabrication technologies and novel arquitectures. In particular, NeuriCam is currently pursuing:

- A version of the processor with deeper weight memory and a 50 MHz operating frequency.
- A 128-neuron version of the processor with 256-byte deep weight memory and a performance level of 5 GCPS.

- On-chip support for discrete-time recurrent neural networks will be provided to allow the efficient implementation of recognition systems for time varying signals such as those encountered in speech recognition systems.
- Software modules capable both of hiding the hardware details from the user and providing a higher level of abstraction in the description of the computational blocks.
- Processor boards and integrated circuits incorporating a CPU for stand-alone embedded applications.
- A parallel neurocomputer based on industry-standard CPUs and the CompactPCI bus.

4 Conclusions

ANNs have a distinct advantage over traditional algorithmical methods when it comes to problems dealing with fuzzy logic, pattern recognition and simulation. For a number of years some software solutions implementing MLP networks have been tried, but its slowness rendered them unusable for most real-time applications.

Dedicated hardware, on the other hand, can prove in the foreseeable future to be a reliable and efficient solution to some of the problems the computer industry is now facing. The traditional approach has not been too much successful in some areas as speech, finger-print and face recognition, and these are notoriously promising areas of business.

NeuriCam's solution is particularly interesting. Selling the ANN as a coprocessor allows for a low-cost, efficient system, and one that loses none of the general capabilities associated with having a PC microprocessor. The fact that NeuriCam bundles the PCI board with software for Linux and Windows that allows the end user to create, customize and use the processor without having to know how it operates in depth is proof that the company is seriously trying to make the idea work.

References

[1] Danese, G., Leporati, F., Ramat, S.: A Parallel Neural Processor for Real-Time Applications. IEEE Magazine, (May-June 2002) 20-31.

[2] NeuriCam: NC3001 Totem – A Digital Processor for Neural Networks. NeuriCam, www.neuricam.com, (1998).

[3] NeuriCam: NC3001 Totem PCI Board. NeuriCam, www.neuricam.com, (1998).

[4] NeuriCam: A Primer on Artificial Neural Networks. NeuriCam, www.neuricam.com, (1998).

[5] Pacific Northwest National Laboratory: What is an Artificial Neural Network?, www.emsl.pnl.gov:2080/proj/neuron/neural/what.html, (1997).

[6] Battiti, R., Tecchiolli, G.: The Reactive Tabu Search. In ORSA Journal on Computing, Vol. 6, n. 2, (1994) 126-140.