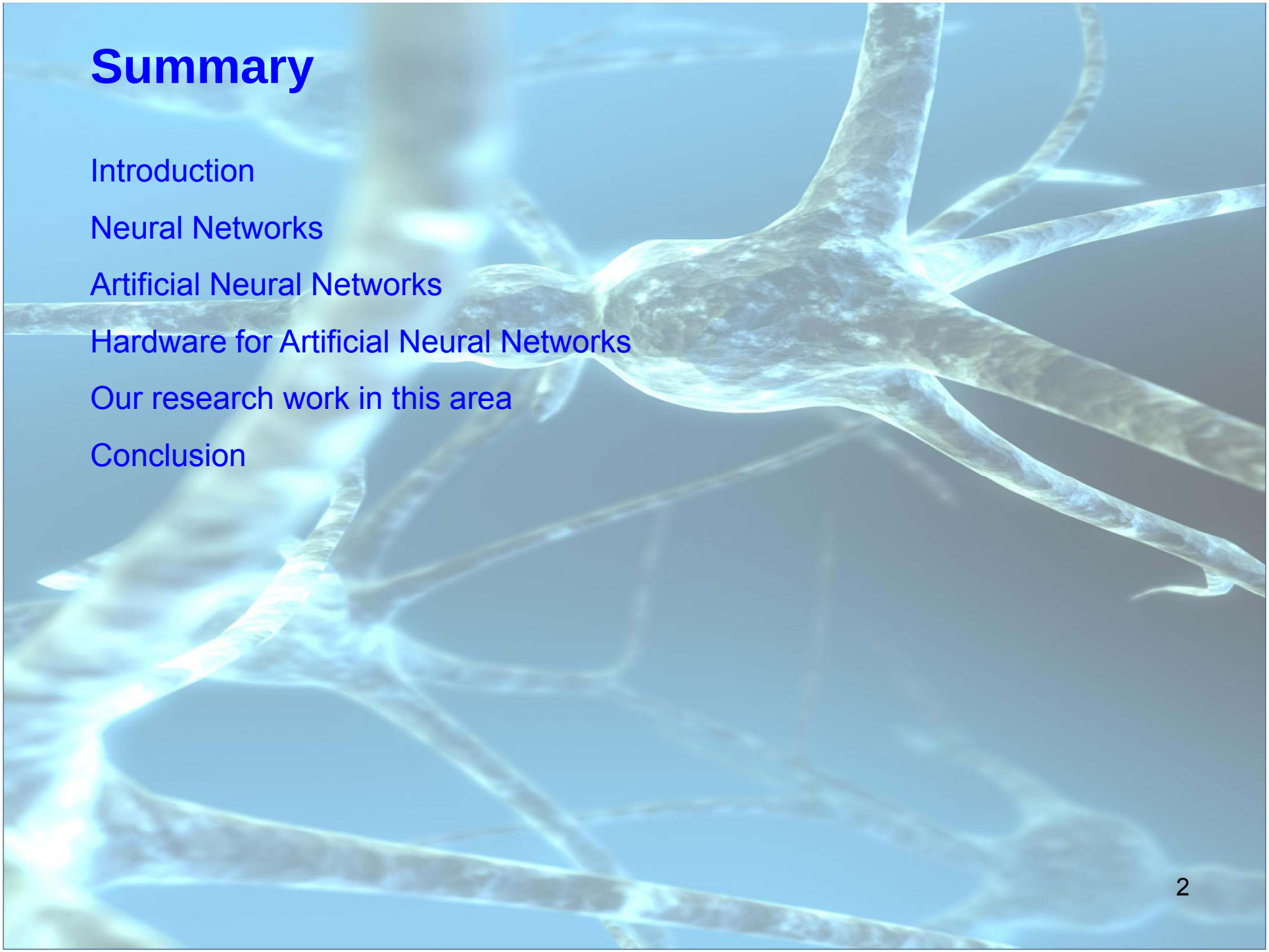
Artificial Neural Networks

Towards a general purpose hardware implementation solution

Morgado Dias

University of Madeira

Madeira Interactive Technologies Institute



Summary

Introduction

Neural Networks

Artificial Neural Networks

Hardware for Artificial Neural Networks

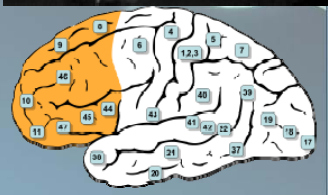
Our research work in this area

Conclusion

Introduction

Brain study is quite recent in spite of the development of other medical areas. Some relevant examples for ANN are:

- Egas Moniz, neurologist and diplomat; first Portuguese Nobel prize(1949). Developed leucotomy (or lobotomy), which consists of cutting the connections to and from the prefrontal cortex. Lobotomy is today considered inhumane and is no longer used but it was an advance at the time (showed the use of certain brain areas).



- Alan Hodgkin and Andrew Huxley, medicine Nobel prize winners in 1963, conducted several tests with the giant axon of squids by applying electric pulses. These test led to the Hodgkin-Huxley model for natural neurons.

-António Damásio describes in “Descartes´ Error” one example of behaviour change as a result of a work accident that severed the brain in the XIXth century.

Introduction

•In 1989 a DARPA report estimated that the largest capacity computer at the time would have, at most, the capacity to simulate the functioning of the brain of a fly.

•Animals present a larger neural development in childhood. One test with a young cat whose eye was covered during early development showed that the cat never fully developed vision from that eye.

•Recently it was possible to recover partial vision for patients which were not born blind through the use of CMOS sensors. The sensor is similar to a camera but it can only be used if the brain was trained to process image. The level of vision recovered was of 10%.

•The processing speed of a natural neuron is 1000 times smaller than the one obtained in electronic circuits.

•In general, the brain is still pretty much UNKNOWN!

Introduction

•Many attempts to mimic the brain have been built mostly because of the tasks that the brain performs well and are difficult to a computer.

•Examples of such tasks are: face recognition, state of mind identification, language processing.

•Artificial Neural Networks attempt to mimic natural neural networks

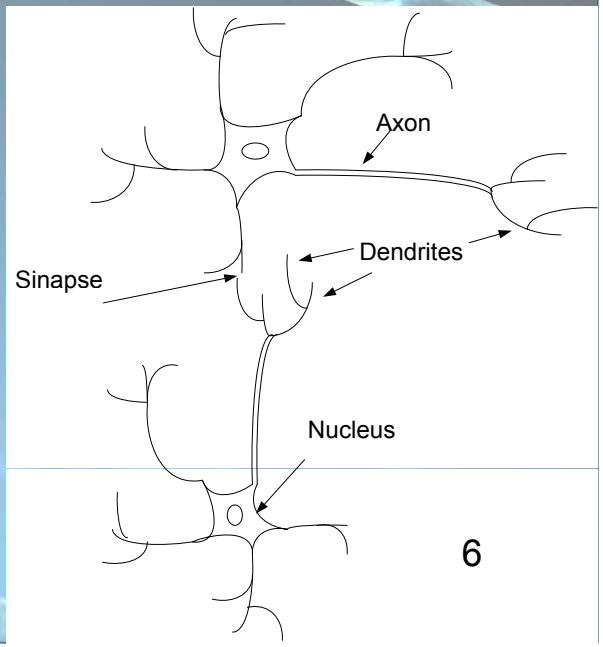
Neural Networks

• The Brain is composed very simple processing units.

• The human brain is composed of about 10 bilion neurons with about 10.000 billion synapses. Each neuron has, on average, 1000 synapses (connections) though some might have up to 6000.

• The neurons, functionally, are grouped in neural networks ant it is common that they share neurons with other networks.

•Neurons communicate through spikes as a result of certain events.



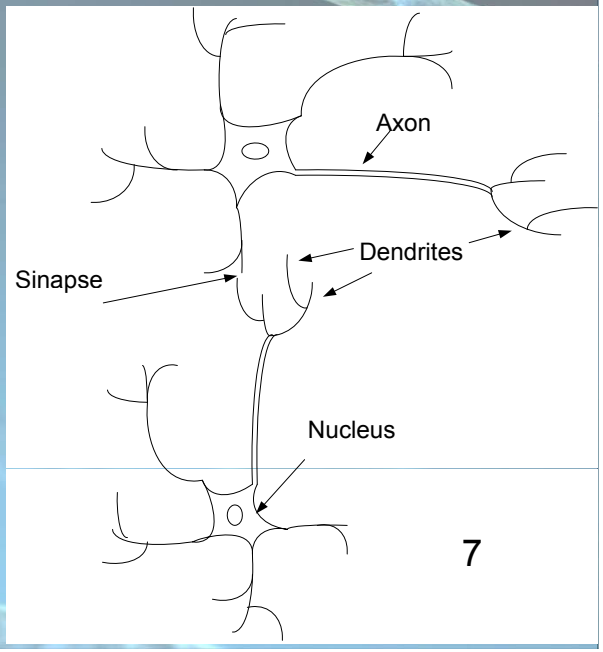
Neural Networks

• The dendrites form a ramified network of nervous fibers with the capacity to transport electric signals to the nucleus.

• The nucleus performs the functions of adding up the inputs affected by their weights and applies a threshold function.

•The axon is a nervous fiber that takes the output signal to other neurons.

•The contact between the axon of one neuron and the dendrite of another neuron is the synapse.



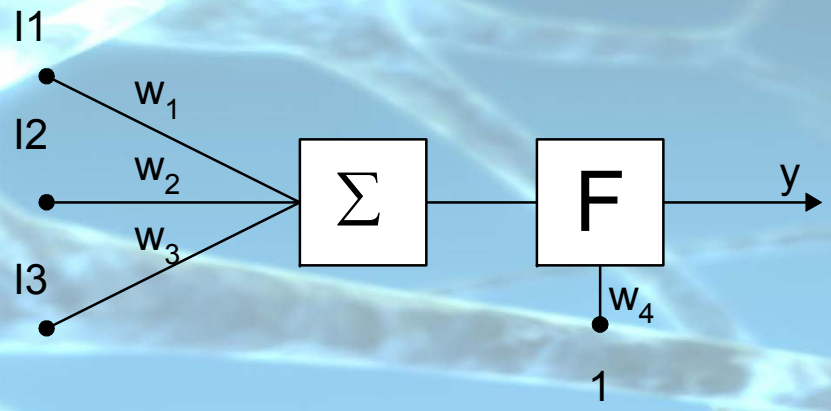
Artificial Neural Networks

• There are many different types of ANN, with different characteristics, all biologically inspired, though some remotely.

• The origin of so many different models is related with the reduced knowledge of the brain functioning.

• Probably the most well known model is the Perceptron or Feedforward Neural Network. This model holds no lateral or feedback connections.

• The neuron performs a very simple function:



$$y = F(\sum_{i=1}^{n} Ii.wi)$$

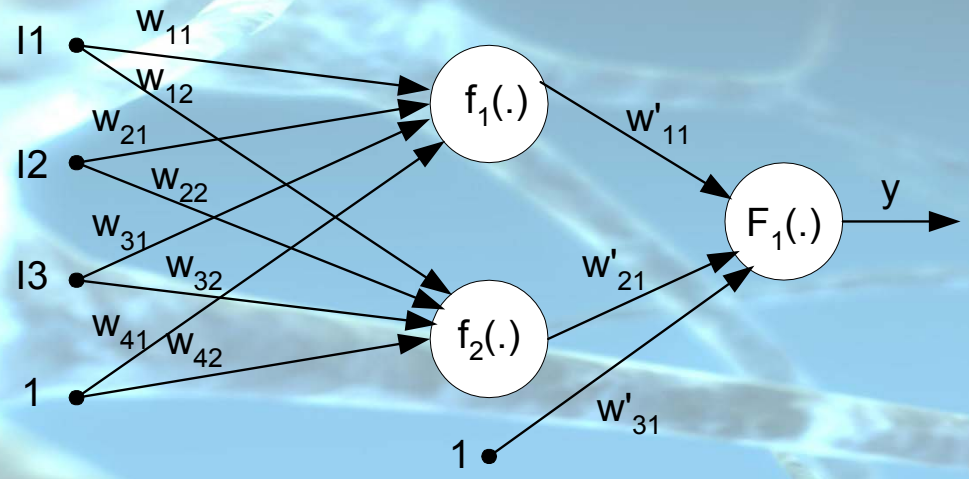
Artificial Neural Networks

• A network of such neurons implements the following function:

$$y = F_1\left(\sum_{j=1}^{nh} w'_{j1}\, f_j\left(\sum_{l=1}^{nI} w_{lj} I_l\right)\right)$$

• One of the advantages of this function is the possibility to choose different activation functions, some of which are nonlinear.

• Although one can use ANN whith any number of layers, it has been proved that one hidden layer ANN is an universal approximator.



Nome	Função	Função
Heaviside	x<0 → y=0 x≥0 → y=1	
Heaviside simétrico	x<0 → y=-1 x≥0 → y=1	
Linear	y=x	
Linear com saturação	x<0 → y=0 0<x<1 → y=x x>1 → y=1	
Linear simétrico com saturação	x<-1 → y=0 -1<x<1 → y=x x>1 → y=1	
Log-sigmoidal	$y = \frac{1}{1+e^{-x}}$	
Tangente hiperbólica	$y = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	

Artificial Neural Networks

• How do they learn?

• Just as with their natural counterpart, they learn through examples that lead to connection (weight) changes.

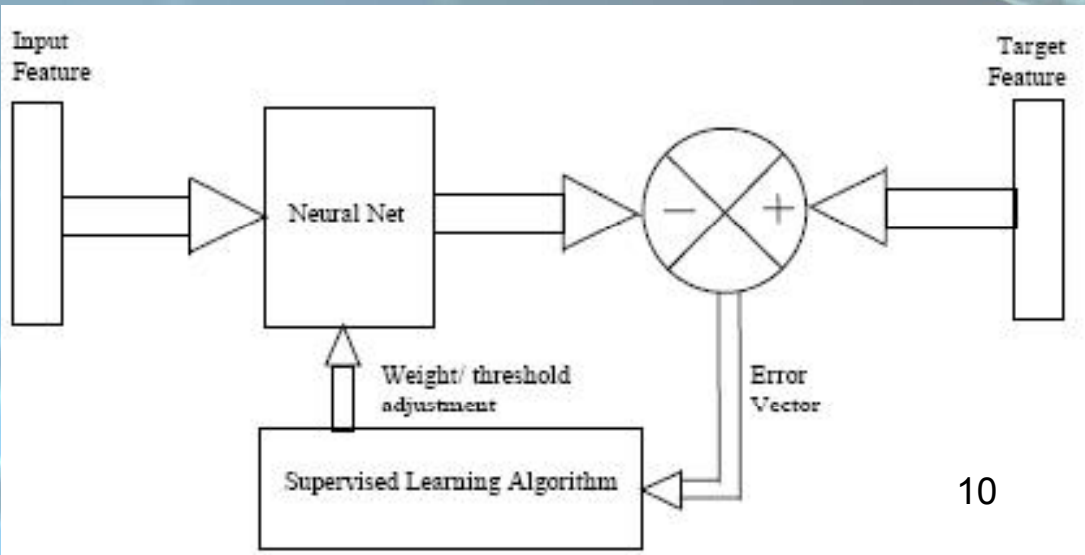
• Although there are many different algorithms, the common part is to use the output error to adjust the weights throughout the ANN to correct its answer.

• ANN weights are updated in an iterative way. Starting with an initial guess, they are, usually, updated in the following way:

$$w_{k+1} = w_k + \alpha_k . p_k$$

Where pk stands for the search direction and αk represents the learning rate.

Different algorithms choose different ways to select αk and pk.



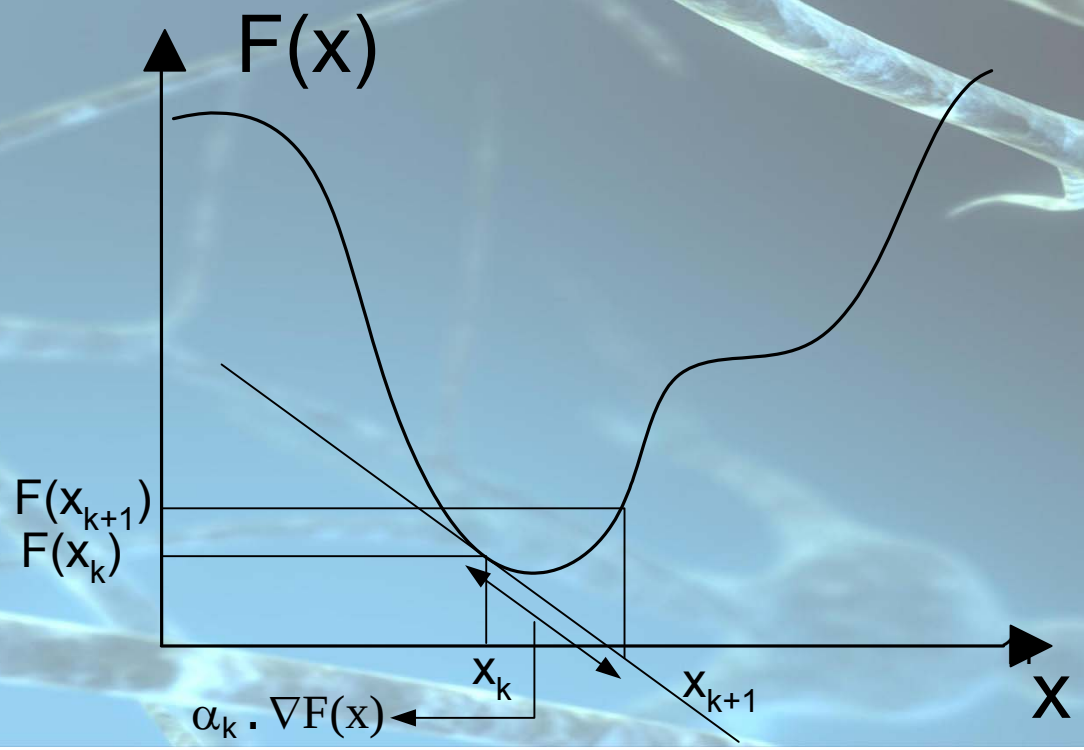
Artificial Neural Networks

• Since the functions to minimize are usually quadratic (squared error), we use derivatives to find the minimum value:

$$w_{k+1} = w_k - \alpha_k . G(x,k)$$

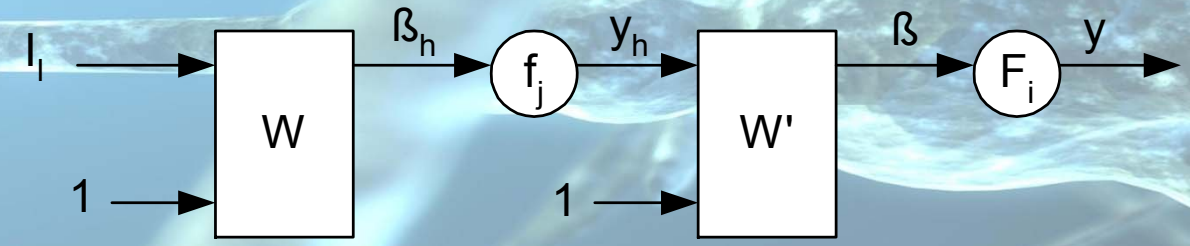
where G(x,k) is the gradient of the function to minimize at iteration k.

• The choice of learning rate is also an important factor:



Artificial Neural Networks

• The previous transparencies explain the principle of updating the weights but not how to distinguish them inside the ANN. This is done through Backpropagation which is based on the derivative chain rule:



$$w(k+1) = w(k) - \alpha . \frac{\partial EQ}{\partial w}$$

$$y = F(\sum_{j=1}^{nn} w'_{jl} . f_j(\sum_{l=1}^{ne} w_{lj} . I_l))$$

$$\frac{\partial EQ}{\partial W'} = \frac{\partial EQ}{\partial y} . \frac{\partial y}{\partial W'} = \frac{\partial EQ}{\partial y} . \frac{\partial y}{\partial \beta} . \frac{\partial \beta}{\partial W'}$$

$$\frac{\partial EQ}{\partial W} = \frac{\partial EQ}{\partial y} . \frac{\partial y}{\partial \beta} . \frac{\partial \beta}{\partial y_h} . \frac{\partial y_h}{\partial \beta_h} . \frac{\partial \beta_h}{\partial W}$$

Artificial Neural Networks

•These algorithms are based on the availability of examples (or a tutor that classifies situations in good or bad). This is similar to human being learning.

•One of the problems of ANN learning (also similar for human beings) concerns the ability to generalize. When we teach a ANN to behave in a certain way, has it learned to solve that particular problem or can it generalize and answer correctly for similar situations?

•To verify this situations there specific methods which usually include using different training and test sequences.

•Once the model is built it can be used to model the behavior of the system, to control it or to do classification (among other tasks).

Hardware for Artificial Neural Networks

Motivation:

Speed – some applications have high speed requirements

Price – some applications cannot afford the price of attaching a PC

Fault tolerance – some applications need fault tolerance in order to obtain graceful degradation instead of complete failure

Impossibility to use a PC – some applications cannot use a PC because of size or the necessity to connect to a living being)

Hardware for Artificial Neural Networks

Most common limitations:

Oversimplified activation functions – too few linear sections, very simple polynomials, …

High error – low accuracy and simplified functions results in high error

Slow calculations – processing can be done in a serial way using few hardware elements, but it becomes very slow

Floating versus fixed point notation – fixed point notation is less resource consuming, floating point can represent larger ranges for the same number of bits and achieve higher precision

Hardware for Artificial Neural Networks

Possible platforms:

GPU – can be used to increase speed of an ANN but cannot be used when there are physical limitations

ASIC – have price and development time limitations. It is only an affordable solution if a very large number of copies is produced

Microprocessors – speed limitations

FPGAs – exhibits a good balance between price and speed. Has limitations regarding implementing non-linear functions

Our research work in this area

Goal: Build a general purpose hardware implementation solution for ANN that can:

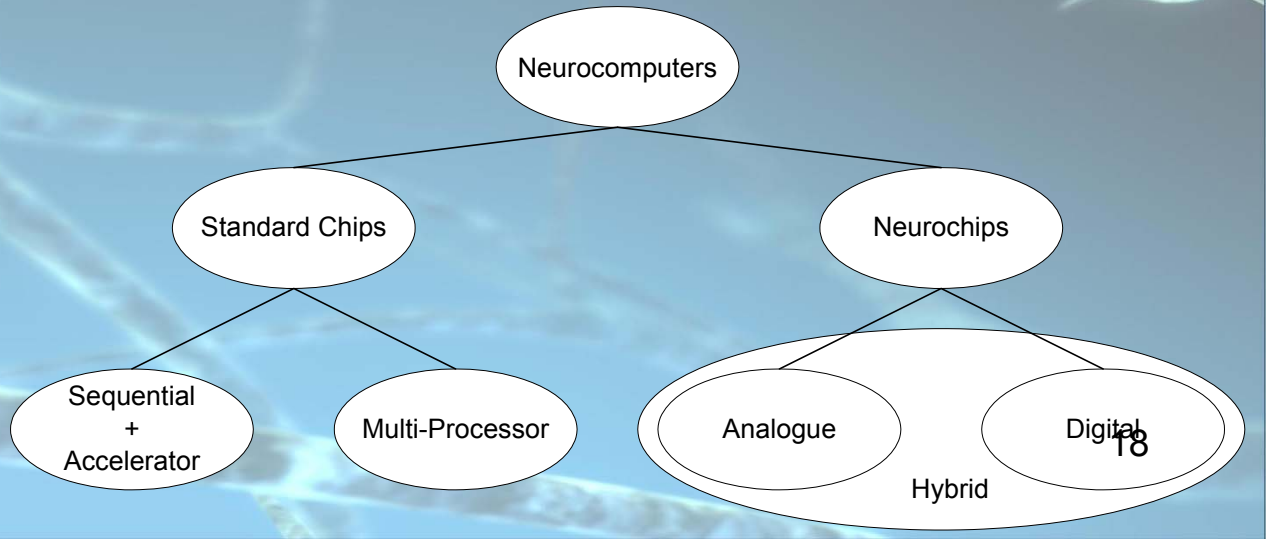
- Directly use the ANN trained in a high level tool
- Introduce the smallest error possible (in the general calculations and the activation functions)
- Be used by users that have low or no hardware knowledge
- Allow the user to make choices to achieve a rational use of the available hardware

Our research work in this area

Step 1 – Study the existing solutions:
Artificial Neural Networks: a Review of Commercial Hardware, Morgado Dias, Ana Antunes, Alexandre Mota, Engineering Applications of Artificial Intelligence, IFAC, 2004.

This work concentrated in the commercial neurochips. 23 hardware solutions were found, although some were are no longer available:

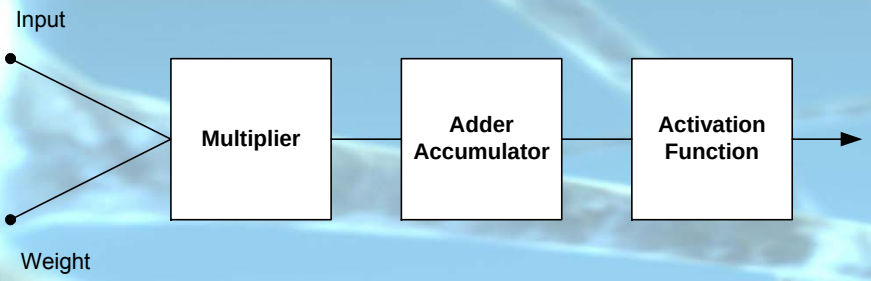
- 17 digital solutions: 4 slice architectures (building blocks); 7 SIMD (single instruction multiple data); 1 Systolic Array (each processor performs one step of the calculation forming a pipeline.); 3 RBF; 2 Other Chips.
- 2 analogue solutions.
- 4 hybrid.

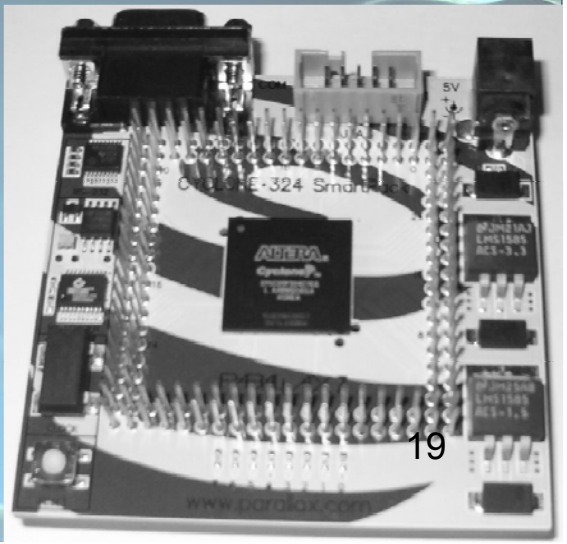


Our research work in this area

Step 2 – Build a simple FPGA implementation:
A high bit resolution FPGA implementation of a FNN with a new algorithm for the activation function, Pedro Ferreira, Pedro Ribeiro, Ana Antunes, Morgado Dias, Neurocomputing, 2007 (original work from 2004).

-Activation function implemented in a LUT.
-Use of symmetry for the hyperbolic tangent.
-Implements a piecewise linear solution that can be tailored to a maximum error definition (supplying the desired maximum error, the number of linear sections and their values is calculated).
-Because of the capacity of the FPGA available it implements the minimal hardware to build the ANN with 32 bits floating point notation.
-Maximum error of $5x10^{-8}$ obtained in a control loop simulation compared to a Matlab implementation.

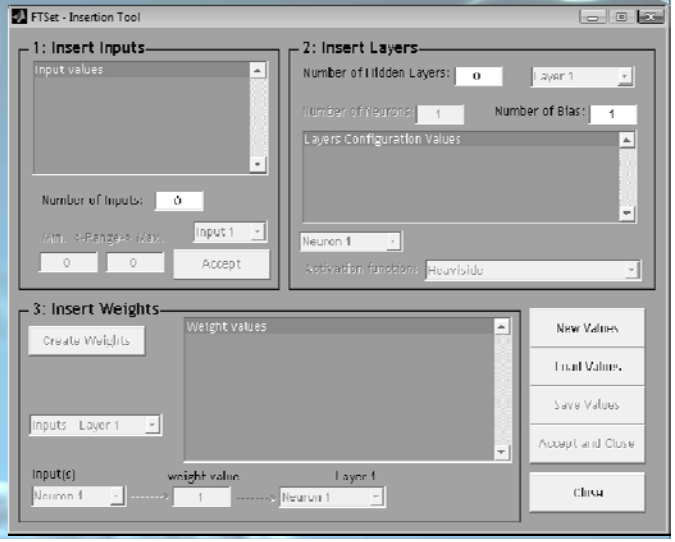




Our research work in this area

Step 3 – Investigate ANN Fault Tolerance/ Graceful degradation:
"FTSET-A Software Tool for Fault Tolerance Evaluation and Improvement", F. Morgado Dias, Rui Borralho, Pedro Fontes, Ana Antunes, Neural Computing and Applications, 2010 (work from 2008).

FTSET tool evaluates and improves the fault tolerance of a given ANN:
-Discusses faults in ANN hardware
-Can work with the hardware limits
-Changes the weights, inputs and the number of neurons to obtain higher fault tolerance
-Stops when predefined values of FT, limits of the hardware (inputs or neurons) or maximum number of iterations is reached.



Our research work in this area

Step 4 – An alternative implementation for the hyperbolic tangent

"Hyperbolic tangent implementation in hardware: a new solution using polynomial modeling of the fractional exponential part", Ivo Nascimento, Ricardo Jardim and Morgado Dias, Neural Computing and Applications, accepted March 2012.

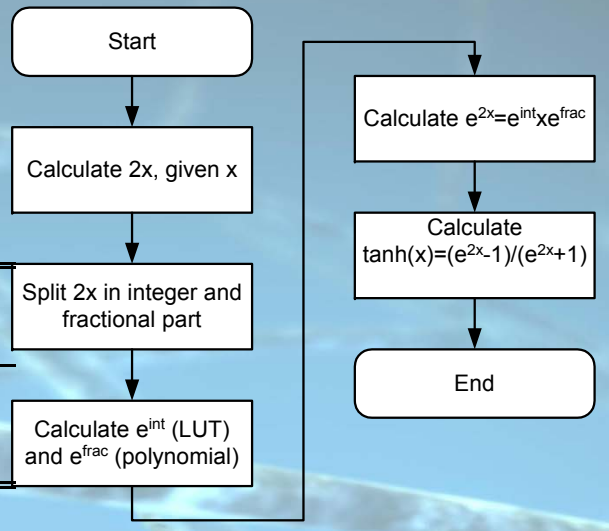
Main idea: split the exponential in integer and fractional part: $e^{x} = e^{x\,\mathrm{int}} \times e^{xfrac}$

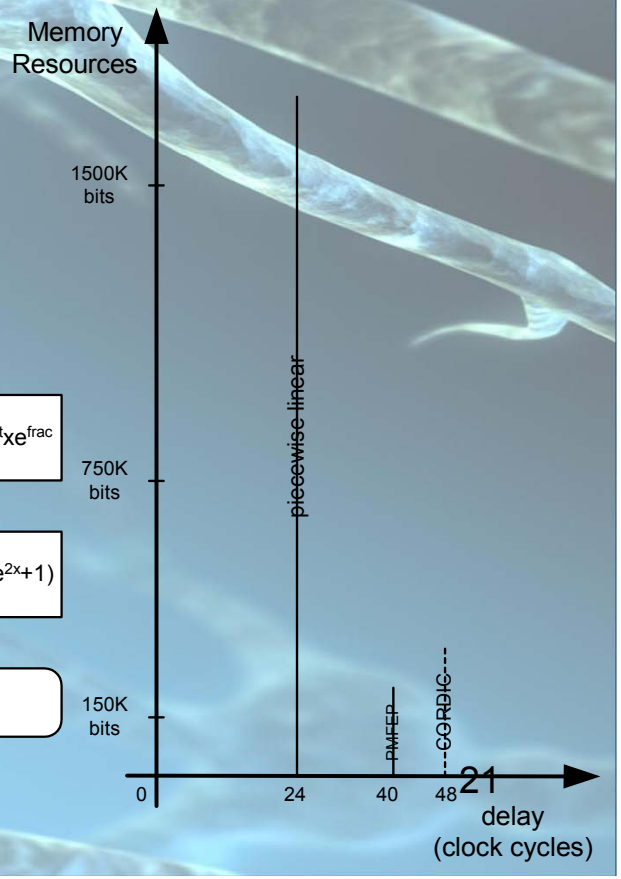
-Use a LUT for the integer part

-Approximate the fractional part with a polynomial

-Compared to our piecewise linear solution, the 10000 sample error improved.

Solution	Error Largest Positive	Error Largest Negative	MSE
Piecewise Linear	0.0432	-0.0432	$2.7475*10^{-4}$
Algorithm PMFEP	0.0188	-0.0098	$5.1396*10^{-5}$

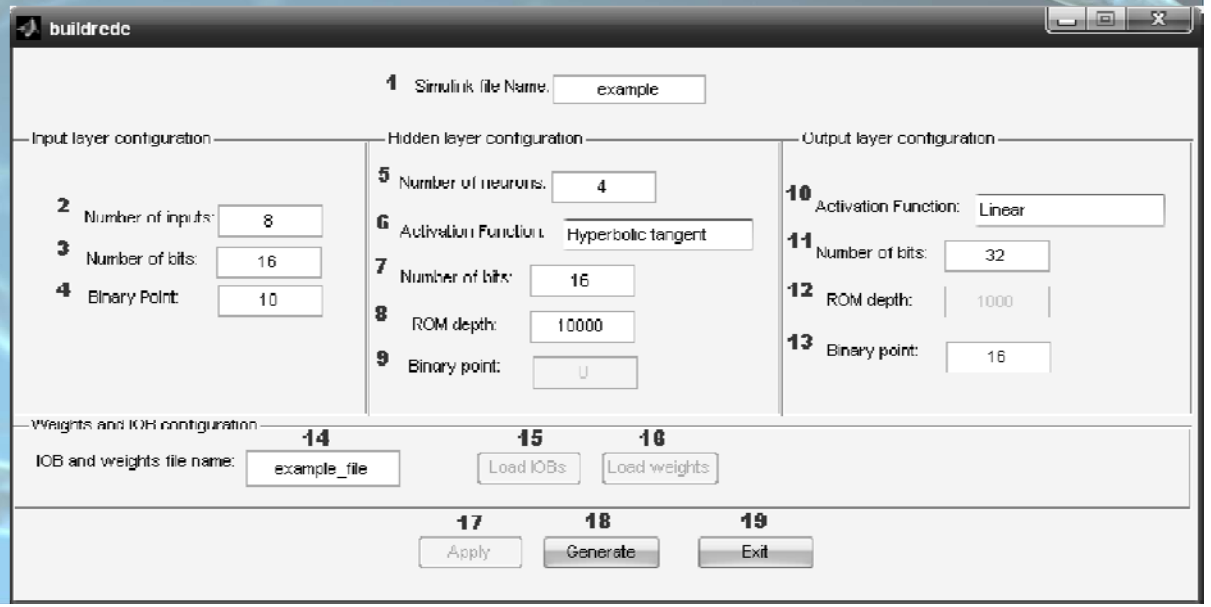




Our research work in this area

Step 5 – Automatic Generation of Neural Hardware – ANGE I
"A Software Tool for Automatic Generation of Neural Hardware", L. Reis, L. Aguiar, F. D. Baptista and Morgado-Dias, International Arab Journal of Information Technology, accepted September 2012.

ANGE I has the following features:
-Builds simulink models that use system generator to create a hardware implementation
-Has only LUT version of the hyperbolic tangent
-Allows selecting the number of bits used for inputs, outputs and LUT
-Produces the code to configure an FPGA with the selected options
-Fixed Point notation



Our research work in this area

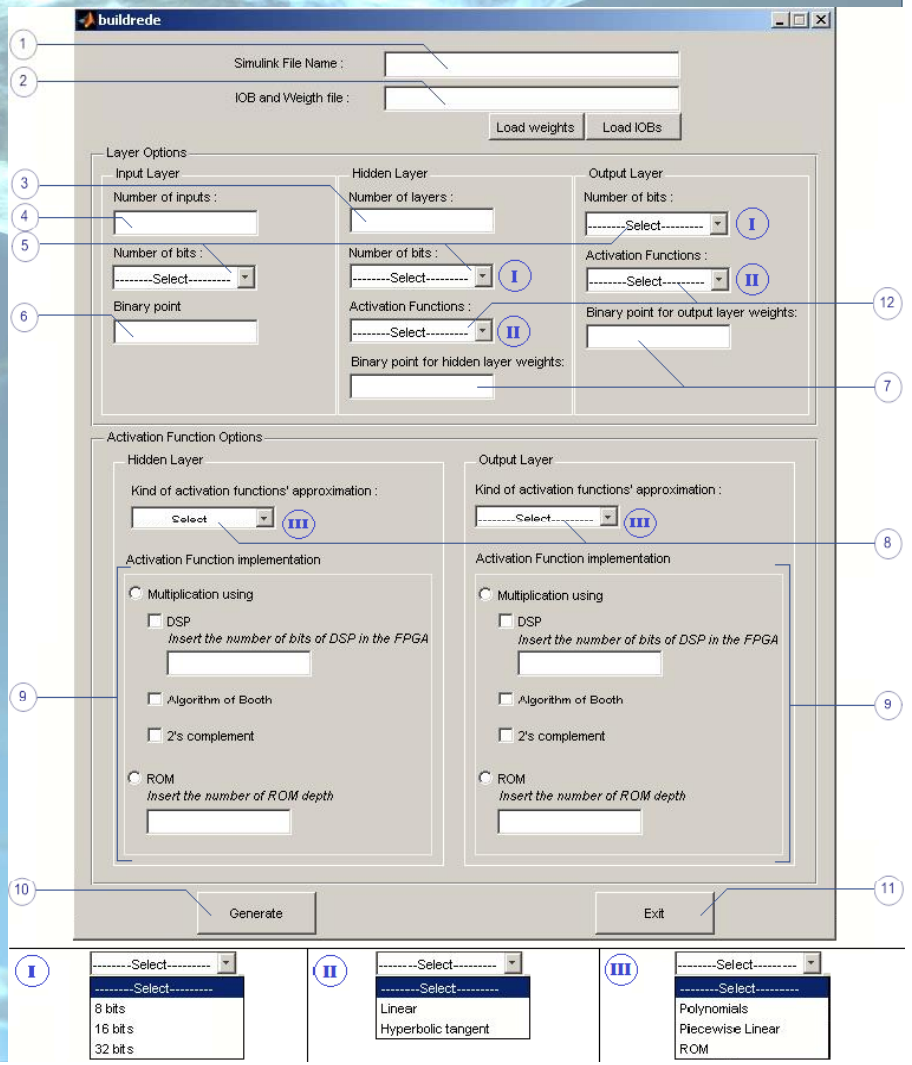
Step 6 – Automatic Generation of Neural Hardware – ANGE II

We are currently developing ANGE II with the following features:

-Allows choosing multiplication type: 2's complement, Booth's algorithm, built in multipliers or DSPs

-Hyperbolic tangent implemented with LUT, polynomial sections (25 third order polynomials obtained by Chebyshev approximation with maximum error of $1{,}929 \times 10^{-7}$), piecewise linear sections

-Optimized to use a minimal number of multiplications



Conclusion

ANN are used in many different fields of application. Some of these fields need hardware implementations because of size, price, fault tolerance or the necessity to connect to living beings.

We are searching for solutions that allow automatic implementation of high level trained ANN.

Our largest step towards that solution is ANGE II which introduces very small error when compared with Matlab, supplies several alternatives for the implementation and can be used with almost no hardware knowledge.

More relevant publications

“Low-resource hardware implementation of the hyperbolic tangent for artificial neural networks”, F. D. Baptista and F. Morgado-Dias, Neural Computing and Applications, accepted April 2013.

“A Hyperbolic Tangent Replacement by Third Order Polynomial Approximation”, Darío Baptista, Fernando Morgado-Dias, Controlo’2012, Funchal, Portugal, July 2012.

“On the Implementation of Different Hyperbolic Tangent Solutions in FPGA”, Darío Baptista, Fernando Morgado-Dias, Controlo’2012, Funchal, Portugal, July 2012.

“A Software Tool for Automatic Generation of Neural Hardware”, L. Reis, L. Aguiar, F. D. Baptista e F. Morgado-Dias, International Arab Journal of Information Technology, accepted September 2012.

“Hyperbolic tangent implementation in hardware: a new solution using polynomial modeling of the fractional exponential part”, Ivo Nascimento, Ricardo Jardim and Morgado Dias, Neural Computing and Applications, accepted March 2012.

“FTSET-A Software Tool for Fault Tolerance Evaluation and Improvement”, F. Morgado Dias, Rui Borralho, Pedro Fontes, Ana Antunes, Neural Computing and Applications, Vol. 19, Number 5, pp.701-712, 2010.

“A high bit resolution FPGA implementation of a FNN with a new algorithm for the activation function”, Pedro Ferreira, Pedro Ribeiro, Ana Antunes, Fernando Morgado, Neurocomputing, Vol. 71, Issues 1-3, pp. 71-77, 2007.

“Artificial Neural Networks: a Review of Commercial Hardware”, Fernando Morgado Dias, Ana Antunes, Alexandre Mota, Engineering Applications of Artificial Intelligence, IFAC, Vol. 17/8, pp. 945-952, 2004.