Suresh Gyan Vihar University, Jaipur
International Journal of Converging Technologies and Management (IJCTM)
Volume 3, Issue 1, 2017
ISSN : 2455-7528

Implementation of Neural Network Model For Cancer Detection Based On Back Propagation With The Help of Python Based Hardware Description Language

Pankaj kumar Sharma Mtech. Research scholar(VLSI),Suresh Gyan vihar university,jaipur(Raj)India

Mr. Ravishanker Sharma Associate Professor(CS dept). Suresh Gyan vihar university,jaipur(Raj)India

Abstract—conventionally the design method of Artificial Neural Networks using VHDL and implement in FPGA is done. VHDL is a programming language that has been designed and optimized for describing the behavior of digital systems. Back propagation algorithm for the design of a neuron is used. Over the last years many improvement strategies have been developed to speed up designing. The neuron is used in the design and implementation of a neural network using FPGA. This Article is based on idea that hardware description has its own unique requirements. My HDL is an open source platform for using python a general purpose high level language for hardware design. A designer using this software can benefit from the power of python language and free open source software as well.

***Keywords*— ANN,VHDL,Field Programmable Gate Array (FPGA),Python,myHDL**

INTRODUCTION :

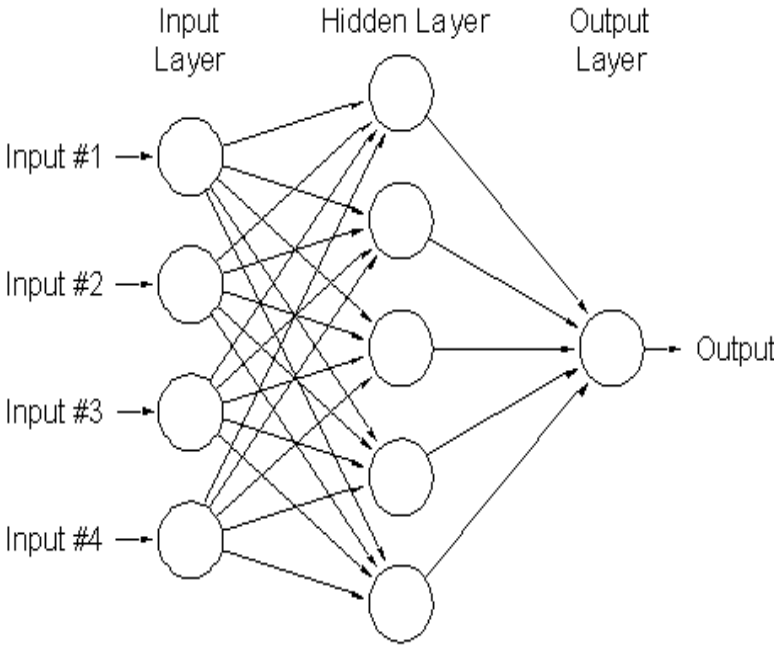
Cancer is a dreadful disease. Millions of people died every year because of this disease. It is very essential for medical practitioners to opt a proper treatment for cancer patients. Therefore cancer cells should be identified correctly. Neural networks are currently a burning research area in medical science, especially in the areas of cardiology, radiology, oncology, urology and etc.

Artificial Neural Network

An artificial neural network (ANN) or commonly as Neural Network (NN) is an interconnected group of artificial neurons that uses a mathematical or computational model for information processing based on a connectionist approach to computation. In most cases an ANN is an adaptive system that changes its structure based on external or internal information that flows through the network. In more practical terms neural networks are non-linear statistical data modeling tools. They can be used to model complex relationships between inputs and outputs or to find patterns in data.

Suresh Gyan Vihar University, Jaipur
International Journal of Converging Technologies and Management (IJCTM)
Volume 3, Issue 1, 2016
ISSN : 2455-7528

Artificial neural networks (ANN) are parallel algorithms. Their inherent parallelism makes them particularly suited to parallel VLSI implementations, i.e. the development of dedicated circuits or architectures that are able to perform many operations in parallel. As ANN involve similar operations to be performed in parallel, it is particularly easy to develop dedicated parallel architectures: most of them are based on the repetition of identical devices, each of them performing one of the operations.



Parallel architectures for ANN are justified by the fact that most ANN learning algorithms involve a high computational load. Who never waited days (of computing time) for the learning of a Multi-Layer Perceptron does not know what neural networks are! Since the early beginning of the neural networks research, one has naturally been tempted to develop specialized machines, in order to lighten the computational load.

Another nice feature of ANN is that most algorithms mainly involve simple operations. Indeed, those algorithms with biological inspiration naturally use the same type of operations as found in biological cells.

Here we present the classification of benign and malignant tumor based on Feed Forward Neural Network trained with Resilient Back Propagation Algorithm

Back Propagation Neural Network (BPNN) algorithm

In recent years, the use of Artificial Neural Networks (ANN), in particular, three layer neural networks, which can be trained to approximate virtually any continuous function, is the most usual type of Multilayer Perceptron (MLP). Multilayer Perceptron propagates the information from the input towards the output layer. MLP has an input layer of neurons, a number of hidden layers, and an output layer. By using sufficient number of hidden neurons in hidden layers MLP can map any nonlinear input–output function to an arbitrary degree of accuracy.

All layers are fully connected and of the feed forward type. The outputs are nonlinear function of inputs, and are controlled by applying weights to the data that are computed. In recent years, the use three layers feed forward neural networks, is the most usual type of feed forward NN, which propagates the information from the input towards the output layer. In addition to these methods, a heuristic optimization algorithm is used to increase the success and speed of these methods. PSO as a heuristic optimization method is successfully applied to train MLPNN. It is proposed to update network weights by reasons of easy implementation and realization, the small number of parameters to be set, and capable of treatment with real numbers, no derivative information. In this study, in order to improve the ability of conventional neural network to escape from a local optimum, the PSO algorithm was used to modify the Network parameter and precision.

RPROP - stands for '**Resilient Propagation**' and is an efficient new learning scheme that performs a direct adaptation of the weight step based on local gradient information. In crucial difference to previously developed adaptation techniques, the effort of adaptation is not blurred by gradient behavior whatsoever. To achieve this, we

Suresh Gyan Vihar University, Jaipur
International Journal of Converging Technologies and Management (IJCTM)
Volume 3, Issue 1, 2016
ISSN : 2455-7528

introduce for each weight its individual update-value Aij, which solely determines the size of the weight-update. This adaptive update-value evolves during the learning process based on its local sight on the error function E, according to the following learning-rule:

$$\Delta_{ij}^{(t)} = \begin{cases} \eta^+ * \Delta_{ij}^{(t-1)} & , \text{ if } \frac{\partial E}{\partial w_{ij}}^{(t-1)} * \frac{\partial E}{\partial w_{ij}}^{(t)} > 0 \\ \eta^- * \Delta_{ij}^{(t-1)} & , \text{ if } \frac{\partial E}{\partial w_{ij}}^{(t-1)} * \frac{\partial E}{\partial w_{ij}}^{(t)} < 0 \\ \Delta_{ij}^{(t-1)} & , \text{ else} \end{cases}$$

$$\text{where } 0 < \eta^- < 1 < \eta^+$$

Verbalized, the adaptation-rule works as follows: Every time the partial derivative of the corresponding weight wij changes its sign, which indicates that the last update was too big and the algorithm has jumped over a local minimum, the update-value Aij is decreased by the factor q-. If the derivative retains its sign, the update-value is slightly increased in order to accelerate convergence in shallow regions. Once the update-value for each weight is adapted, the weight-update itself follows a very simple rule: if the derivative is positive (increasing error), the weight is de- creased by its update-value, if the derivative is negative, the update-value is added

$$\Delta w_{ij}^{(t)} = \begin{cases} -\Delta_{ij}^{(t)} & , \text{ if } \frac{\partial E}{\partial w_{ij}}^{(t)} > 0 \\ +\Delta_{ij}^{(t)} & , \text{ if } \frac{\partial E}{\partial w_{ij}}^{(t)} < 0 \\ 0 & , \text{ else} \end{cases}$$

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} + \Delta w_{ij}^{(t)}$$

However, there is one exception: If the partial derivative changes sign, i.e. the previous step was too large and the minimum was missed, the previous weight-update is reverted:

$$\Delta w_{ij}^{(t)} = -\Delta w_{ij}^{(t-1)}, \text{ if } \frac{\partial E}{\partial w_{ij}}^{(t-1)} * \frac{\partial E}{\partial w_{ij}}^{(t)} < 0$$

Due to that 'backtracking' weight-step, the derivative is supposed to change its sign once again in the following step. In order to avoid a double punishment of the update- value, there should be no adaptation of the update-value in the succeeding step. In practice this can be done by setting $\frac{\partial E}{\partial w_{ij}}^{(t-1)} := 0$ in the

Adaption rule above. The update-values and the weights are changed every time the whole pattern set has been presented once to the network (learning by epoch).

Artificial neural networks (ANN) have found widespread deployment in a broad spectrum of classification, perception, association and control applications. Any kind of standard data can be categorized by using the hardware implementation. Artificial neural networks (ANNs) have been mostly implemented in software. This has benefits; since the designer does not need to

know the inner workings of neural network elements, but can concentrate on the application of the neural network. However, a disadvantage in real-time applications of software-based ANNs is slower execution compared with hardware-based ANNs. Most of the data or applications are based on a Real-time. Hardware is more susceptible than software implementation. In Artificial Neural Network (ANN) MATLAB is used for software implementation and VHDL is used for Hardware implementation mostly. Using MATLAB we find out the weights of the standardized data which is taken from net. By using this calculated weights and inputs from standardized data we can categorize the standardized data. The simulated waveform are observed in active HDL and implemented in FPGA. The Network is designed and trained in software using MATLAB Neural Network processing toolbox. Once network is trained, correct weights are determined, it has to hard coded on FPGA. The VHDL code is compiled, synthesized and implemented in Quartus II.

Here we have presented the implementation of neural networks by FPGAs. The proposed network architecture is modular, being possible to easily increase or decrease the number of neurons as well as layers. FPGAs can be used for portable, modular, and reconfigurable hardware solutions for neural networks, which have been mostly used to be realized on computers until now.

Proposed myHDL design-

Main functions of the tool:

High level modeling: python is a very high level language, famous for giving elegant solutions to complex modeling problems. Besides this the ability of python of rapid application development and experimentation is outstanding .All these features of MyHDL make it an ideal solution for hardware modeling.

In order to model hardware concurrency, MyHDL makes use of python generators which are used in MyHDL resembles the always block in verilog and processes in VHDL. The software models a hardware module as a function that returns generators. A wide range of features, such as arbitrary hierarchy, named port association, arrays for instances and conditional instantiation, can be easily supported because of this straightforward approach. Classes in myhdl support bit oriented operations and a class for enumeration types.

Converting the design to Verilog and VHDL:

Although there are some limitation, designs in myHDL can be converted into Verilog and VHDL. But the convertible subset is restricted .it is much wider than the standard synthesis subset. it includes features that can be used for high level modeling and test benches . Also, the conversion limitations are only applicable for the code inside the operators .python's full power can be used without compromising convertibility outside the generators. The conversion provides a path into the traditional design flow, including synthesis and implementation.

A number of complex tasks in VHDL and Verilog .such as handling of signed arithmetic issues, can be directly automated with the help of converter in myhdl .the converter works on a fully elaborate, instantiated design. As a result the original design structure can be arbitrarily complex.

Simulations and verification functions: MyHDL has a built in simulator that runs on top of python interpreter, which supports waveform viewing. Tracing the changes of the signal in a vcd file enables viewing of waveform .the software enables the python unit test framework to be used in hardware designs.

Unit testing is a very popular verification in modern software engineering .but when it comes to hardware platform, testing is not that common .in case of verilog designs ,we can also use myhdl as

hardware verification language by co- simulation with traditional HDL simulators.

How MYHDL benefit Research scholars and Professionals:

1. Free and open source - A research scholar will least interested in purchasing a costly software .MyHDL is a well documented open source software which a research scholar can easily use.
2. Simplicity of python- Choice of right developing language makes the task easier .Among high level programming languages, python is one of the simplest to learn, and MyHDL aims to bring the same level of simplicity in hardware design. The automated conversion of design into verilog/VHDL also eases the job of a designer as a beginner.
3. Latest software developments readily available for the designer – As MyHDL is based on python language, the developments which are going on in software methodologies which lead to measure improvements in developments process ,are directly available to the designer.
4. Use of dynamic languages directly for hardware design- dynamic languages like python ,perl ,tcl play important role in modern digital designs ,many developers prefer these languages to HDL languages ,because they can get things quickly done with these
5. Unified algorithm and implementation –usually development of an algorithm and implementation of a design are done by different engineers in various platforms .engineers for algorithm development commonly use python, which hdl implementation are done on another platform.

   myHDL unifies these two domains .both can be developed in the same environment, even though

   Conversion from algorithm to HDL development requires knowledge of HDL based design, being able to do this in the powerful environment significantly increases the quality and productivity of design .The designer can directly reuse the verification work in both domains.

Disadvantages :

MYHDL can not completely substitute the use of other HDL languages. If a designer wants to have accurate timing simulations needs to go for industry standards ,VHDL and Verilog.

References:

[1] Rafid Ahmed Khalil, "Hardware Implementation of Backpropagation Neural Networks on Field programmable Gate Array (FPGA)", Al-Rafidain Engineering, Vol.16, No.3, Aug.2008.

[2] Aydoğan Savran, Serkan Ünsal," Hardware implementation of a feed forward neural network using FPGA".

[3] Aguiar L., Reis L., and Morgado-Dias F., "Neuron Implementation Using System Generator," in Proceedings of the 9th Portuguese Conference on Automatic Control, Portugal, 2010.

[4] Hassanien Ella Aboul and Ali H.M. Jafar, ―Rough set approach for generation of classification rules of Breast Cancer data,Journal Informatica, vol. 15, pp. 23–38, 2004.

[5] Jamarani S. M. h., Behnam H. and Rezairad G. A., ―Multiwavelet Based Neural Network for Breast Cancer Diagnosisl, GVIP 05 Conference, pp. 19-21, 2005.

[6]. Waserman - "Neural computing". Wayne Wolf – "Modern VLSI Design".

[7]. Jai Sachit Paul., "A Python Based Hardware Discription Language",Electronics For You ,Feb 2015.

[8]. B. Magesh et al., "Computer Aided Diagnosis System for the Identification and Classification of Lessions in Lungs", International Journal of Computer Trends and Technology, ISSN : 2231-2803, IJCTT May-June 2011.