IMPROVING OF ARTIFICIAL NEURAL NETWORKS PERFORMANCE BY USING GPU'S: A SURVEY

Israel Tabarez-Paz[1], Neil Hernández-Gress[2] and Miguel González Mendoza[2].

[1]Universidad Autónoma del Estado de México
Blvd. Universitario s/n, Predio San Javier Atizapán de Zaragoza,
México {itabarezp}@uaemex.mx
http://www.uaem.mx/cuyuaps/vallemexico.html
[2]Tecnológico de Monterrey, Campus Estado de México,
Carretera Lago de Guadalupe km 3.5 Atizapán de Zaragoza Col. Margarita Maza de
Juarez, Atizapán de Zaragora, México
{ngress, mgonza}@itesm.mxhttp://www.itesm.edu

ABSTRACT

In this paper we study the improvement in the performance of Artificial Neural Networks (ANN) by using parallel programming in GPU or FPGA architectures. It is well known that ANN can be parallelized according to particular characteristics of the training algorithm. We discuss both approaches: the software (GPU) and the Hardware (FPGA). Different training strategies are discussed: the Perceptron training unit, the Support Vector Machines (SVM) and Spiking Neural Networks (SNN). The different approaches are evaluated by the training speed and performance. On the other hand, algorithms were coded by authors in the hardware, like Nvidia card, FPGA or sequential circuits that depends on methodology used, to compare learning time with between GPU and CPU. Also, the main applications were made for recognition pattern, like acoustic speech, odor and clustering According to literature, GPU has a great advantage compared to CPU, this in the learning time except when it implies rendering of images, despite several architectures of Nvidia cards and CPU's. Also, in the survey we introduce a brief description of the types of ANN and its techniques of execution to be related with results of researching.

KEYWORDS

GPU, FPGA, Artificial Neural Networks, Spiking Neural Networks, Support Vector Machines.

1. INTRODUCTION

This paper presents a survey of the improvement of training time and performance of Artificial Neural Networks (ANN). Some trends of topics are about ANN is Parallel Programming to solve problems such as clustering (Herrero-Lopez [8]), pattern recognition (Olaf [28]), regression (Carpenter [19]), building of ANN in a specific hardware, such as FPGAs (Papadonikolakis [7]).

Historically, ANN were developed in second half of century XX as a result of research made in Second War World. First generations are MC Culloc (1943), Hebb (1949), Perceptron (1957), Adaline y Madaline (1959), Cognitron (1980) y Hopfield (1982). Then, in 1982 Hinton and William developed the known algorithm called back propagation. This can be considered as second generation of ANN. The third generation was studied since 1930, with British scientists Alan Lloyd Hodgkin (1914-1998) y Andrew Fielding Huxley (he was born in 1917), nowadays SNN are based on this model. These authors finished their research in 1950 and their mathematical model is known as Hodgkin – Huxley [27]

The perceptron is the first training unit in which a training algorithm for linearly separable problems, that consist of a single neuron. Mathematically, itis represented by anstraight line equation [36]. However, non – linear problems can't be solved with this methodology.

However, in decade of 1990 was started the development of SVM [35], even this methodology had been invented since 1979 [32] by Vapnik, to solve more complex problems, linearlly separable or non – separable. In this case, support vectors are a set of vectors placed in border of clusters for classifying or densities detection. An advantage related to this methodology was that scientist calculated its architecture that consists of three layers; however, SVM consume many memory resources.

In 2003 was developed the methodology SNN also called Spike Response Model (SRM) (Bohte [29], Olaf[28]). At the same time, Izhikevich [30] developed a reduced model of Hodgkin – Huxley model that consist on two differential equations that explain behavior of mammal neurons. The main diference between SRM and Izhikevich's model are the differential equations.

In case of Hodgkin – Huxley model has four differential equations with partial non linear derivatives, and depends on the space and time. This model describes propagation and generation of potential of a big axon of squid in order to explain the main properties. SNN's are based on the model described in last paragraph, because is the model most similar to the neurons of mammals [28]

Information in the mammalian neurons of the brain, is coded with spikes, around 55 mV. A spike is an electrical pulse along its axon. The similar Artificial Neural Network with mammalian neuron is Spiking Neural (SNN), what sends a response according to the data encoded in the time, so it is more suited for applications where the timing of input signals carries important information (e.g., speech recognition and other signal-processing applications). Also, SNN can be applied to the same problems that depend on behavior of time of parameters because of its singular characteristic of coding in the time [13]

Respect to parallel programming [38] calculations are carried out simultaneously, operating on the principle that large problems can often be divided into smaller ones, that are solved concurrently. There are several different forms of parallel computing: bit-level, instruction level, data, and task parallelism. This manuscript is focused on instruction level and task. In the figure 1, we can see principle of parallel computing.

Problems of parallel programming can be solved in OpenGL, Cg, C, C++ and Fortran.□□ Finally, they are developed for CUDA language.

This paper is distributed as follows: in section 2 the state of the art of ANN is presented, in section 3 the performance of ANN in parallel programming is included, finally in section 4 are the conclusions.

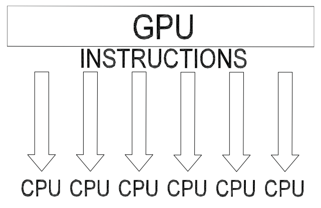

Figure 1. Parallel programming.

2. Relating Works on The Use of Parallel Programming in Artificial Neural Networks

Recently works are shown in the following paragraph. According to Sridhar [1], proposes application of a technology that consists of a chip thermal modelling based on Neural Network and GPU's. Learning time is compared between CPU and GPU for this application. As a conclusion, GPU is faster then CPU.

Table 1. State of the Art Parallel.

AUTHOR	CONTRIBUTION	YEAR
Arash [2]	Implements Izhikevich's model in a GPU of Spiking Neural Network. Comparisons between CPU and GPU are shown and considerable speedup is achievable with the approach depending on the system architecture and the SNN complexity.	2011
Lowe [3]	He uses several GPU's of Nvidia to compare an Artificial Neural Network with eight neurons in the hide layer about its training time. As a result, Nvidia GTX is faster than other cards that he used. However, he doesn't say which is the model of Artificial Neural Network.	2011
Sergio Herrero [8]	Compares learning time and precision between two GPU targets and libraries of MatLab.	2010
Izhikevich [9]	Designs a hybrid model for SNN in order to combine continuous and discontinuous numeric methods.	2010
Bhuiyan [11]	Compares the models of SNN such as Izhikevich and Hodgkin Huxley, these models applied to recognition of characters.	2010
Yudanov [13]	Implements a hybrid method with numeric integration of Parker Sochacki (PS) with adaptative order. This is validated at the moment in the comparision made between GPU and CPU in their characteristics.	2010
Scanzio [12]	He compares the speed of processing in CUDA of algorithms feed fordward and back propagation.	2010
Xin Jin [14]	He Applies a chip called SpiNNaker that contains several processors ARM968 that has a speed of processing of 200MHz each one. He compares the results usingMatLab. He evaluate the state of actualization of the neuron, arrangement of entry, processing of a new entry;. However he don't experiment with other databases.	2010
Papadonikolakis [15]	He has focused on improving the speed of learning and efficiency of SVMs using several methods. Also, he compares these parameters between a GPU and a FPGA programming Gilbert' algorithm.	2009
Nageswaran [17]	He presents a compilation of theIzhikevich's models.	2009

Thomas andLuk [18]	The author proposes a Simulation with maximum 1024 neurons in a FPGA the Izhikevich's model.	2009
Stewart andBair [20]	He Applies Runge – Kutta's method for Izhikevich and Hodgkin Huxle's models. As a result, second method is more efficient.	2009
Prabhu [21]	He applies GPU for pattern classifier in images. He focuses on the degree of parallelism of a problem. He uses maximum size of image of 256 MB, and in a video memory GPU of 768 MB. As a result, author compares Dual – Core AMD processor with a Geforce 6150 GPU, and when the number of patterns increase, the CPU is linearly slower than GPU, But when the network size increase the curve isn't linear.	2008
Catanzaro [22]	The author proposes theSVM using algorithm of Sequential Minimal Optimization (SMO), also he compares time of learning and precision of classification between GPU and libraries of SVM of MatLab.	2008
Martínez [23]	He implementes Fuzzy ART Neural Network in a GPU in order to compare results with CPU processor. Type of data applied are images in format RGB. Also the author uses dual – core Pentium 4 at 3.2 GHz, although the Neural Network is only accelerated in a GeForce 7800 GT card. As a conclusion, CPU is faster in training stage that CPU, in spite of GPU is faster in testing stage.	2007
Philipp [24]	He programs Spiking Neural Network in a FPGA. About results, author says that this hardware can be simulated through thousands of neurons, however author does not show conclusions about it. On the other side, he focuses on synchronization of nodes according to the frequency of the signal.	2007
Pavlidis, et. al [26]	He Applies evolution algorithms using SNN, however according to author efficiency of the network was not calculated.	2005
Zhongwen [25]	He calculates the learning time in a multilayer perceptron in a CPU and a GPU, spending 11328 ms (CPU) and 46 ms (GPU) respectively.	2005
Olaf [28]	The author Shows theory about SNN and the problem of codification of the entries.	2004
Bohte [29]	He also made a comparison with different algorithms as Spikeprop. The work does not show the learning time. On the other hand, he presents differences between traditional ANN and SNN.	2003
Izhikevich [30]	He simulates in MatLab the Hodgkin – Huxley's model. Also, he gots to execute maximum 10000 neurons and 1000000 of sinapsys.	2003

3. Parallel Programming in ANNs

3.1. Types of architectures for parallelization

Artificial Neural Networks can be parallelized in a specific hardware, for example GPU, FPGA such a Thomas [18], sequential circuits, or specific card [14] However, according to the most often used, we have focused on GPU and FPGA.

In the case of SVM's, they are better designed in a GPUbecause the optimizing methodrequires of solving repetitively operations of matrixes. However, for SNN methodology is better to use

FPGA or sequential circuits, although it can be simulated in GPU. The reason is that in a GPU thelearning time is calculated with a non linear and exponential equation,which algebraic order depends directly of number of neurons in the input. This equation is solved with mathematical approximations, so if the number of neurons in a input layer is increasing, the computational complexity too. However, to design SNN architecture in the circuit mentioned, the time does not need a mathematical solution to find the threshold, this is only detected with a simplex circuit.

3.2. Configuration of a GPU

This section is focused in program CUDA in a GPU. Because of each author uses his own resources, then there is not a standardhardware of interest for comparing the results with other researchers, however results of other investigations can be used as a reference to improve the algorithms.

One of the biggest problems to start isto configure the hardware selected. Some ANNs need to be programmed with kernel in three dimensions, such as SNN. The simplest configuration in one or two dimensions is perceptron method. Parameters to be considered in hardware are threads, blocks and grids. In case of perceptron, each column of a block can be used as a layer;in contrast for SNN each block can represent only one neuron. Sometimes computational resources are not enough. Other problem to be considerated is thatthis method is recursive and weightsmust be frequently adjusted because of local memory is not enough.

However, all details of programming depend on the ability to program each design it can be easy or not. Also, efficiency can be better if the researcher knowswhen a specific methodology could be used. But the most important depends on the researcher's ability to decide when and how much can be parallelized (the grade of parallelism).

The hardware configuration is selected in each kernel, the number of threads and the number of blocks per grid are chosen [38]

3.3. Grade of parallelization

All ANNs can be parallelized in several levels. First, perceptron can be parallelized per layer. In other words, all outputs of neuronsper layer can be calculated at the same time because of thesimple form of its activation function. On the other hand, each layer must be calculated sequentially because of input of hidden layer depends on output of previous layer. However, circular buffer, this procedure can be designed, through FIFO principle (First in – First out).So, when all the output values of the first iteration are known all the layers could be parallelized. However, this can be implemented in other research.

Second, in case of SVM, is similar to perceptron but the difficulty increases when the quantity of parameters increases to calculate the dimension in where the problem is separable. So, a handle matrix of more than one columnis needed and requirements of computational resources are bigger. In this case, we kernels in CUDA with more than one dimension could be needed.

Third, for SNN an implement algorithmof a mathematical or sequential method to calculate the value of threshold in amplitude and time is needed. This impliesthatmany values of time are needed, as well as, maybe hundreds or thousands per row in a block. So, a neuron is represented as a grid in tree dimensions, where each row can represent a previous neuron to be added for one output of the following neuron. This method requiresgood memory resources.

Therefore, sometimes it is not recommended to parallel more than hardware features allow you to do.

3.4. Examples of parallelization in ANNs

Parallelization of ANN consists in two phases: first learning phase, and second execution phase. Parallelization for learning phase in case of SVM is in parallelization of matrixes operations, what is well defined in webpage of CUDA [37] although this depends on optimization method applied what is focused on solving quadratic problem [38]. In case of SNN, the mathematical calculus in phase of learning is more sequential because of its approximation method.But in both SVM and SNN all weight of hidden layers neurons can be parallelized and calculated at the same time.

The second phase refers about to evaluations of weights calculated according to inputs. The sections A, B and C show a way of parallelizing of this phase.

Preceptron. Graphically, the following components of the model represent the actual activity of a neuron. All inputs are summed altogether and modified by the weights. This activity is referred as a linear combination. Finally, an activation function controls the amplitude of the output. This process is described in the figure 2. Each thread represents a layer. In the GPU, parallelism for a neuron of perceptron is focused on mathematical operations [38]

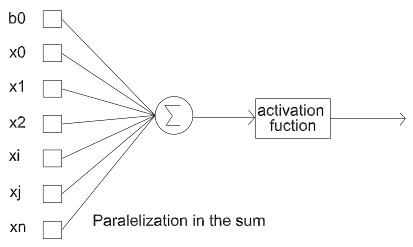

Figure 2. Simple perceptron.

Backpropagation. In this case has the samealgorithm in weight thatSNN. The weight needs to be adjusted per iteration. Parallelization in two dimensionsis like the figure 3.

Support Vector Machines (SVM). In the figure 3, the parallelization of SVM could be observed, in this case each thread represents a layer with n or k number of rows, however the mathematics calculus are parallelized into of each neuron of hidden layer in other kernel, what is solved as a optimization problem. The dimension of the Hessian matrix isequal to numberof input parameters. So, the multiplication of matrix is another operation to parallelize that can be solved in a separated kernel. In the figure 3, the y0 neuron gets at the same time all values multiplied per its respective weight, the in other kernel in CUDA the mathematical operations are parallelized. At the same time the other neurons of hidden layer computing its respective output. However, the next layer cannot calculate its output without the previous layer has done.

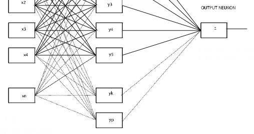

Figure 3. Parallelization of Support Vector Machine (SVM), 2D Array.

Spiking Neural Networks (SNN). Thearrange of figure 4 represents a configuration of the GPU device in three dimensions. This is a solution for parallelizing SNNalgorithm.The cube showedin this figure is only a neuron of a hidden or output layer.There are as many cubes as neurons are required. Each cube is divided in blocks, what depend on the length of time in the input [28]. All neurons per layer can be calculated in parallel, but a disadvantage is thatthis procedure requires many resources of memory.

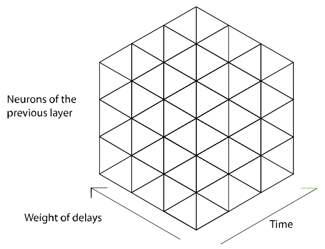

Figure 4. Parallelization of Spiking Neural Network (3D Processor Array).

SNN has significant characteristics that must be considerated. The synapses of the biological neuron are modeled as weights. Let’s remember that the synapse of the biological neuron is which interconnects the neural network and gives the strength of the connection. For an artificial neuron, the weight is a number, and represents the synapse. A negative weight reflects an inhibitory connection, while positive values designate excitatory connections.Inherent parallelism of commodity graphic hardware is used to accelerate the computationofANN.According to Nikola [34], taxonomy of parallelization approaches for neurosimulations is represented in the figure 5.

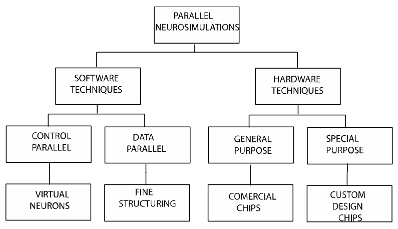

Figure 5. Taxonomy of parallelization approaches for neurosimulations.

Sridhar [1] says that the main advantage of GPU over CPU is high computational parallelism and efficiency with a relatively low cost.However, it is difficult to design an algorithm. Also, the author says that although exist Integrated Circuits (IC) for high parallelism, it is very difficult to translate this parallelism in an efficient software.On the other hand, human brain can be trained to solve complex problems, such as thermal modeling of specific IC layouts.

Prabhu [21] compares efficiency of the human brain with enormous computational powerand parallel environs of GPU’s, so we understand that GPU has some limitations. According to him, the role played by Graphical Processing Unit (GPU) is approaching to Artificial Neural Networks to the nature of human brain. Also, GPU’s have been used for rendering high quality images in real time, virtual reality simulations and games. Modern GPU’s can perform highly intensive parallel tasks.

4. CONCLUSIONS

In this paper weconclude thatparallelismin ANNincrease speedof learning time. However, is very difficult to design this sort of algorithms. On the other hand, we can parallelize by hardware (FPGA) of software (GPU). Tendency is study to know which algorithm is the most efficient and faster, because of their mathematical characteristics and their architecture. So, is better to solve a problem with large database using SVM and SNN than traditional ANN.

The importance to compare the efficiency between these algorithms is to know the error in the results and which is faster for learning according to quantity of instances and parameters per instance. So, with this information is possible to know what applications are the most appropriates for each application.

As a future work, there are some aspects, as parallelizing SVM or SNN in a GPU and SNN in a FPGA, then compare learning time. However, also is necessary to propone an important application to solve real problems.

REFERENCES

[1] Sridhar, A.; Vincenzi, A.; Ruggiero, M.; Atienza, D.; , "Neural Network-Based Thermal Simulation of Integrated Circuits on GPUs," Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on , vol.31, no.1, pp.23-36, Jan. 2012
[2] Ahmadi, Arash; Soleimani, Hamid; , "A GPU based simulation of multilayer spiking neural networks," Electrical Engineering (ICEE), 2011 19th Iranian Conference on , vol., no., pp.1-5, 17-19 May 2011
[3] Lowe, E.W.; Woetzel, N.; Meiler, J.; , "Poster: GPU-accelerated artificial neural network for QSAR modeling," Computational Advances in Bio and Medical Sciences (ICCABS), 2011 IEEE 1st International Conference on , vol., no., pp.254, 3-5 Feb. 2011
[4] B. Kirk, David; W. Hwu, Wen-mai “Programming Massively Parallel Processors” Ed. Morgan Kaufmann, 2010.
[5] Qi Li; Salman, R.; Kecman, V.; , "An intelligent system for accelerating parallel SVM classification problems on large datasets using GPU," Intelligent Systems Design and Applications (ISDA), 2010 10th International Conference on , vol., no., pp.1131-1135, Nov. 29 2010-Dec. 1 2010
[6] Tsung-Kai Lin; Shao-Yi Chien; , "Support Vector Machines on GPU with Sparse Matrix Format," Machine Learning and Applications (ICMLA), 2010 Ninth International Conference on , vol., no., pp.313-318, 12-14 Dec. 2010
[7] Papadonikolakis, M.; Bouganis, C.: "A novel FPGA-based SVM classifier," Field-Programmable Technology (FPT), 2010 International Conference on , vol., no., pp.283-286, 8-10 Dec. 2010
[8] Sergio Herrero-Lopez, John R. Williams, Abel Sanchez,: Parallel Multiclass Classification using SVMs on GPUs, November 2010.
[9] Izhikevich, E.M.: "Hybrid spiking models", vol. 368, issue 1930, pp. 5061-5070, Nov 2010
[10] Venkittaraman Vivek, Pallipuram Krishnamani: ACCELERATION OF SPIKING NEURAL NETWORKS ON SINGLE- GPU AND MULTI-GPU SYSTEMS, ProQuest document ID: 204037751, Publication Number: AAT 147555, May 2010.
[11] Bhuiyan, M.A.; Pallipuram, V.K.; Smith, M.C.: "Acceleration of spiking neural networks in emerging multi-core and GPU architectures," Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on , vol., no., pp.1-8, 19-23 April 2010
[12] Scanzio, S.; Cumani, S.; Gemello, R.; Mana, F.; Laface, P.; , "Parallel implementation of artificial neural network training," Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on , vol., no., pp.4902-4905, 14-19 March 2010.
[13] Yudanov, D.; Shaaban, M.; Melton, R.; Reznik, L.; GPU-Based Simulation of Spiking Neural Networks with Real-Time Performance & High Accuracy, Feb 2010.
[14] XIN JIN: Parallel Simulation of Neural Networks on Spinnaker Universal Neuromorphic Hardware, University of Manchester, 2010
[15] Papadonikolakis, M.; Bouganis, C.-S.; Constantinides, G.: "Performance comparison of GPU and FPGA architectures for the SVM training problem," Field-Programmable Technology, 2009. FPT 2009. International Conferenceon , vol., no., pp.388-391, 9-11 Dec. 2009
[16] Fidjeland, A.K.; Roesch, E.B.; Shanahan, M.P.; Luk, W.; , "NeMo: A Platform for Neural Modelling of Spiking Neurons Using GPUs," Application-specific Systems, Architectures and Processors, 2009. ASAP 2009. 20th IEEE International Conference on , vol., no., pp.137-144, 7-9 July 2009

[17] Nageswaran, J.M., Dutt, N.: Krichmar, J.L.; Nicolau, A.; Veidenbaum, A.; , "Efficient simulation of large-scale Spiking Neural Networks using CUDA graphics processors," Neural Networks, 2009. IJCNN 2009. International JointConferenceon , vol., no., pp.2145-2152, 14-19 June 2009
[18] Thomas, D.B.;Luk, W.: "FPGA Accelerated Simulation of Biologically Plausible Spiking Neural Networks," Field Programmable Custom Computing Machines, 2009. FCCM '09. 17th IEEE Symposium on , vol., no., pp.45-52, 5-7 April 2009
[19] Carpenter, A.; "CUSVM: A CUDA IMPLEMENTATION OF SUPPORT VECTOR CLASSIFICATION AND REGRESSION", Jan. 2009.
[20] Stewart R.D., Bair W.: "Spiking neural network simulation: numerical integration with the Parker-Sochacki method", Jan. 2009
[21] Prabhu, R.D.; "SOMGPU: An unsupervised pattern classifier on Graphical Processing Unit," Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on , vol., no., pp.1011-1018, 1-6 June 2008
[22] Bryan Catanzaro, Narayanan Sundaram, KurtKeutzer: Fast Support Vector Machine Training and Classification on Graphics Processors, International Conference on Machine Learning, Helsinki, Finland, 2008.
[23] Martínez Z. M., Díaz P. F. J., Díez H. J. F, Antón R. M.: Fuzzy ART Neural Network Parallel Computing on the GPU, Springer (2007).
[24] Philipp S., Grübl A., Meier K., Schemmel J.: Interconnecting VLSI Spiking Neural Networks Using Isochronous Connections, Springer (2007).
[25] L. Zhongwen, L. Hongzhi and W. Xincai: Artificial Neural Network Computation on Graphic Process Unit (2005).
[26] N.G. Pavlidis, D.K. Tasoulis, V.P. Plagianakos, G. Nikiforidis, M.N. Vrahatis: Spiking Neural Network Training Using Evolutionary Algorithms, IEEE International Joint Conference on, pp: 2190 – 2194, vol. 4 ,december 2005.
[27] SeijasFossi C., Caralli D' Ambrosio A.: Uso de las máquinas de soportepara la estimación del potencial de acción cellular, Revista de Ingeniería UC. Vol. 11, Nº 1, 56 – 61 (2004).
[28] Olaf Booij,: Temporal Pattern Classification using Spiking Neural Networks, Intelligent Sensory Information Systems Informatics Institute Faculty of Science Universiteit van Amsterdam (2004).
[29] Sander M. B.: Spiking Neural Networks, Universiteit Leiden (2003).
[30] Izhikevich, E.M.: "Simple model of spiking neurons," Neural Networks, IEEE Transactions on , vol.14, no.6, pp. 1569- 1572, Nov. 2003
[31] Sander M. Bohte, Joost N. Kok, Han La Poutre: Error-backpropagation in temporally encoded networks of spiking neurons, Neurocomputing 48, pp: 17 – 37, (2002).
[32] Platt C. J.: Sequiential Minimal Optimization: A fast Algorithm for Tarining Support Vector Machines (1998).
[33] Osuna E., Freund R., Girosi F.: An Improved Training Algorithm for Support Vector Machines, In Proc. of IEEE NNSP'97.
[34] Nikola B. Serbediija: Simulating Artificial Neural Networks on Parallel Architectures (1996).
[35] Vapnik V., Cortes C., Support Vector Networks (1995).
[36] FausetLaurene, Fundamentals of Neural Networks, ARCHITECTURE, ALGORITHMS, AND APPLICATIONS, Prentice Halls, 1994.
[37] A. L. Hodgkin and A. F. Huxley: A quantitative description of membrane current and its application to conduction and excitation in nerve, J. Physiol. (1952).
[38] http://www.nvidia.com
[39] http://www.svms.org

Authors

M en C Israel Tabarez Paz: He works for Universidad Autónoma del Estado de Méxicoas a researcher. He is focus on Artificial Intelligence, Artificial Neural Networks. Also he is a PhD. Student in Computing Science in InstitutoTecnológico y de EstudiosSuperiores de Monterrey Campus Estado de México (ITESM).

PhD. Neil Hernández Gress:He works for de InstitutoTecnológico y de EstudiosSuperiores de Monterrey Campus Estado de México (ITESM)as a Director of Research Technological Development and Postgrade in ITESM. He is focus on Artificial Intelligence, Artificial Neural Networks.

PhD. Miguel González Mendoza.Head of the P Program in Computer Sciences and Engineering in ITESM.Secretary in Mexican Society on Artificial Intelligence.