Journal of Theoretical and Applied Information Technology
30th September 2016. Vol.91. No.2
© 2005 - 2016 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

FACE RECOGNITION USING ARTIFICIAL NEURAL NETWORKS IN PARALLEL ARCHITECTURE

[1]BATYRKHAN OMAROV, [2]AZIZAH SULIMAN, [3]KAISAR KUSHIBAR
[1]PhD candidate of College of Information Technology, Universiti Tenaga Nasional, Kuala Lumpur, Malaysia
[2]Prof. Madya Dr., College of Information Technology, Universiti Tenaga Nasional
[3]Master, University of Burgundy, Le Creusot, France.
E-mail: [1]batyahan@gmail.com, [2]azizah@uniten.edu.my, [3]kukaba.fm@gmail.com

ABSTRACT

Face detection and recognition is the main aspect for different important areas such as video surveillance, biometrics, interactive game applications, human computer interaction and access control systems. These systems require fast real time detection and recognition with high recognition rate. In this paper we propose implementation of the Artificial Neural Network by using high performance computing architecture based on Graphics Processing Unit to get face recognition with high accuracy and more speedup. There, we consider a parallel training approach for backpropagation algorithm for face recognition. For the high performance of face recognition it was used Compute Unified Device Architecture (CUDA) on a GPU.

The experimental results demonstrate a significant decrease on executing times and greater speedup than serial implementation.

Keywords: *Artificial Neural Networks, CUDA, Face recognition, GPU, Parallel Computing.*

1. INTRODUCTION

Nowadays, Face detection and recognition has become far-famed area of image processing and analysis and computer vision research. Mainly we meet such kind of systems everywhere, for example, security systems, social networks, smart phones, etc. [1, 2]. Using and image facial recognition algorithms are proposed to estimate and give decision on where that face is located. In own case, according to the direction each image is classified into several classes as left, right, up and straight.

For some types of such kind of problems using an artificial neural network (ANN) will be one of the most effective methods [3]. In addition, Backpropagation algorithm will serve one of the top ones of commonly used algorithms for ANN learning technique. In this case, backpropagation algorithm is built of parties of interconnected neurons, where each unit has real-valued inputs and returns a single real-valued output. It is commonly used ANN learning method, which is suitable for tasks where the learning target function is defined over specimen that can be described by a predefined features vector or vector of pixel values. Also, output result may be real-valued, discrete-valued or vector of real and discrete valued attributes. Furthermore, training examples may contain some errors and fact assessment of the learned function may be required. All these factors make ANN an effective method in image recognition problem. Some research of particular application meets on [4, 5].

One of the main difficulties of ANN is the significant amount of time needed in the training phase performance to solve complex problems. Depending on grow of number of hidden layers and neurons, the required time to ANN learning process and new instance assessment time grows by leaps and bounds. On the other hand, the rate of successful classification depends on growing of number of hidden layer and neurons. So, in general, the more training instances the network is guaranteed, the more effective result can be achieved. Thus, it is very important to carry out training with a considerable number of neurons in the hidden layer and with a large number of training examples, and with spending relatively low training time.

The statement that each layer makes its own calculation independently from other neurons leads us to any layer parallel calculations which can take place, and a parallel architecture which can be used for this purpose.

1.1. Motivation

Nowadays, facial recognition is an active and actual research area. This is composed by multiple areas as computer vision, pattern recognition, image processing, artificial intelligence, artificial neural networks and computer architectures. There are many applications that use face recognition for security goals as access control or security systems. The system gives good results, mainly when the fingerprint system cannot be used to the recognition goal.

In spite of the reliable methods of identification systems such as fingerprinting or iris scanning, facial recognition is attractive because of the friendly properties used.

It is clearly seen that, there are many challenges on facial recognition problem which are related to reliability and calculative cost, and time cost of the technique. The main goal of the research is improving face recognition using artificial neural network, productivity increasing via the use of large-scale parallel data processing, reached by the implementation of artificial neural network architecture based on Graphic processing unit.

2. RELATED WORKS

With the development of new technologies we have multi-core processors and graphic processing units (GPU) with significant power in our desktop and servers, available to everyone. Using parallel computing techniques for artificial neural network and deep learning has become a budding research area by using current modern hardware.

In [6] authors gives a research survey of the state of the art parallel computer hardware from the perspective of a neural networks user focusing on the performance of ANN on affordable parallel computer hardware such as clusters, multi-core processor machines, workstations and PCs.

Lindholm et al. [7] and Ryoo et al. [8] consider GPU-related problems concerning conventional parallel programs. They study NVIFIA's Tesla unified graphics and

They study developing parallel programs on the basis on CUDA programming API that can be performed based on NVIDIA's Tesla unified graphics and NVIDIA GPUs computing architecture. In their research, they consider CUDA as an extension of C/C++ programming language that developers writing programs call cores. Cores are carried into effect in parallel through a set of threads. Also, they discuss some other approaches concerning to GPU as OpenCL, PGI Accelerator and Brook.

Nickolls et al. [9] and Che et al. [10] treat with CPU based parallel calculation approaches as MPI (Message Passing Interface), OpenMP or Pthreads. Most of the works have to deal with CPU and GPU based parallel computing on a very common level only and not providing an applicable assessment scenario for a specific target.

Jang et al. [11] deals with Multi-Layer Perceptron based (MLP) text detection algorithm performed in OpenMP and CUDA. They tried to simplify the algorithm for all users, even for those users who do not have very good knowledge about programming on GPU. They consider that MLP consists of one input layer and one output layer, also one or more hidden layers. A difficulty of their scheme is describing a short assessment section. In contrast, our research based on Backpropagation algorithm and focuses more specifically on the assessment details. The authors also examined improvements relating the computation times of their application from using parallelization.

Xavier et al. [12] deal with parallel training of Backpropagation using CUDA impleemtation of Basic Linear Algebra Subprogram. They compared implementation on classical CPU and CUDA variation of hidden neurons and comparing the performance of CPU and GPU execution.

Shetal et al. [13] compare CPU in Matlab and GPU implementation for pattern recognition algorithm, concretely, recognize hand written digits.

Summarizing, in our research we examine multi-core versus GPU using parallel computing, on the other hand our work priorities neural network face recognition problem using Backpropagation paradigm by giving specific parameters.

Many methods can be found using parallel computing of neural networks that have been implemented on different architectures. For instance, Pethrick et al. [14] described several approaches for neural network training. In [15] describes the different parallelization levels of neural network. Also, [16] divided Data Parallel category to two types as Structural data parallel category and Topological data parallel category.

This research work focuses on machine learning, parallel computing with high performance, using GPU computing, as it implements algorithm that considerably improves ANN training time as contrasted a sequential face recognition algorithm.

Summing up, in our work on the one hand we analyze the multi-core versus GPU performance gains by parallelization, but on the other hand we also analyze the application behavior of neural network face recognition by a sweep of problem specific parameters. In addition, in our research we implement and run a different number of hidden neurons and cores for processing.

3. GPU ARCHITECTURE AND CUDA PROGRAMMING MODEL

GPUs were designed to perform graphic processes on computers. Nowadays, they have large computing power afforded by thousands of processing units that support fast clock speeds. In the last decade GPUs have been used to solve complicated tasks owing to high-power parallel hardware architecture.

Graphic processors have a huge computation capacity for parallel computing of Image rendering issue. CUDA has C based programming model ensure easy implement in all-purpose computation.

GPU's programmable units are a multiple processor that consists of a set of a single program multiple-data (SIMD) processor. For greater efficiency, GPU processes elements that are called vertices or fragments are used parallel in the identical program. Each element is not dependent on the others, and in programming model elements cannot interrelate with each other. GPU programs structure follows this method: each element is processed parallel by an individual program.

Figure 1 illustrates a CUDA programming and memory model that is based on a parallel compute block called grid. There, an overlook of the threads running inside the device structure is given. It can be seen as parting of host hardware CPU and GPU device, that thread processing organized in blocks is separated into grids of the processing. There, kernels have different grid parameters as its dimensionality. As Figure 1 illustrates, the size of blocks and grid of kernel 1 is different from the one used by kernel 2.

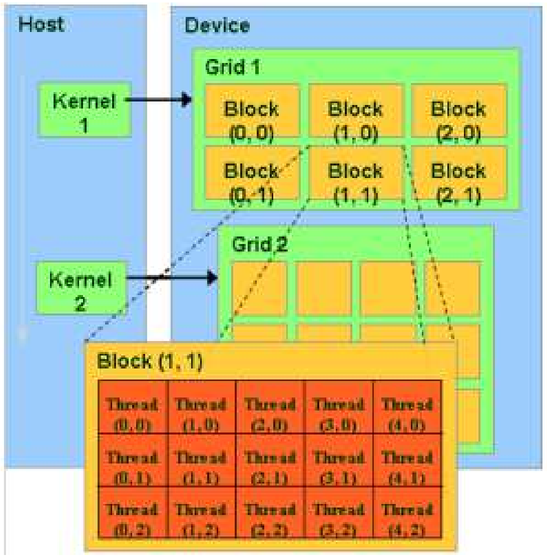

Figure 1. CUDA Programming model (courtesy: NVIDIA)

3.1. CUDA Memory Model

CUDA memory model access rules are given in Figure 2, the detailed explanation is given in Table 1. CUDA architecture is constructed by a scalable array of multi-processors, with each of them having eight scalar processors, a shared memory chip, and a multithreading unit. Multiprocessor creates and manages, also executes parallel threads with diminished cost. The threads are constructed blocks, which are run in a single multiprocessor. In own case, blocks are constructed grids.

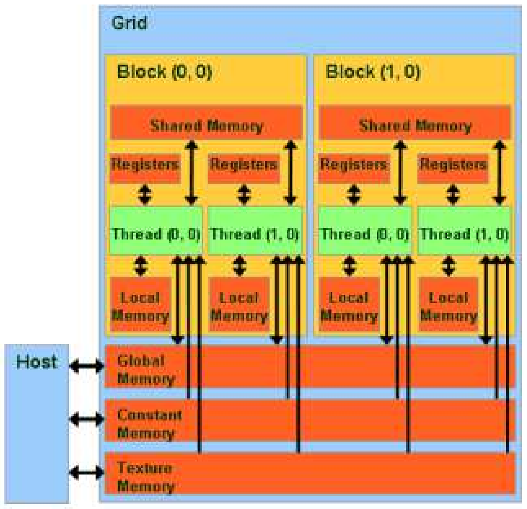

Figure 2. CUDA Memory Model

Program in CUDA calls a grid, and it will be run in the GPU. Blocks are numbered and spread to available multiprocessors. Threads run into the device and have access to memory inside the device. Threads have access to registers and the memory for reading and writing sing memory spaces as a shared memory, a local memory, and a global memory. A shared memory used by the threads, the local memory of the thread, and the global memory of the GPU. Blocks have access to shared memory for reading and writing. Constant and textures memories can be accessed by the grid for reading only.

Table 1. Memory Model Access Rules

Memory space	When accessed	Rule
Register	By thread	Read/Write
Local	By thread	Read/Write
Shared	By block	Read/Write
Global	By grid	Read/Write
Constant	By grid	Read Only
Texture	By grid	Read Only

4. FACE RECOGNITION USING ARTIFICIAL NEURAL NETWORKS IN GPU

4.1 Mathematical Model and Methods

Program in CUDA calls a grid, and it will be run in the GPU

ANN provides a commonly practical method for supervised learning and unsupervised learning functions from given samples. There we use real valued, discrete valued, and vector valued functions. There are some algorithms that allow attuning network parameters to find the best fit training set of input and output parameters. ANN is stable to errors of training data and able to be successfully applied to issues as image processing, speech recognition, and image recognition [3].

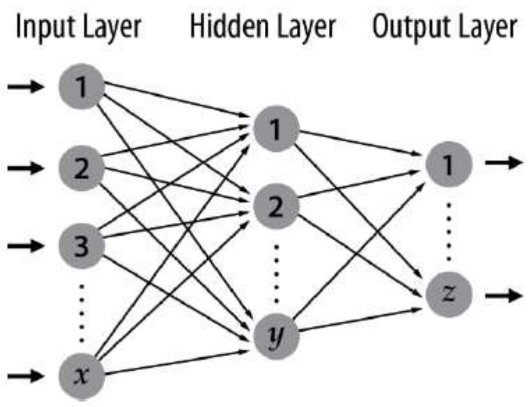

Figure 3. ANN's schema

Figure 3 illustrates general structure of an ANN. ANN work with a set of neurons that connected with each other. Each neuron receives input data, fulfill some linear combinations and return the result, which is the assessment of some function f for the value x=a.

The first step is to modify the backpropagation algorithm so that it can be implemented in a SIMD fashion. In backpropagation algorithm, input patterns are presented to the neural network. Based on the input pattern the network calculates an output pattern. After that the output is compared with a desired pattern, and an error vector is calculated. The computed error will be backpropagated by the network. Based on the error amount on each connection, the weights of the network are changed. After, the network receives next pattern, and the previous procedure is repeated. In the parallel version of the backpropagation, the weights will be stored instead of changing after each template. After the first calculation in several threads, the stored weights are added, and then the weights are updated, depending on the change of total weight are computed.

Let, consider a network which with n output to solve the task with p training patterns. Parallel backpropagation algorithm tries to find a mean squared error for one training. The weights are updated as (1) and (2) formulas.

$$\Delta w_{ij} = \rho \sum_{p=1}^{P} x_j^p act_i^p (1 - act_i^p)(des_i^p - act_i^p) \quad (1)$$

$$\Delta \vartheta_{jk} = \rho \sum_{p=1}^{P} \left[i_k^p x_j^p (1 - x_i^p) \sum_{n-1}^{N} w_{nj} act_n^p (1 - act_n^p)(des_n^p - act_n^p) \right] \quad (2)$$

Where, des_n^p is the desired output and des_n^p is the actual output of the n-th output neuron for the p-th training pattern, $\Delta \vartheta_{jk}, \Delta w_{ij}$ are the first and second stage weights, p is enough small step size, i_k^p and x_j^p are the k-th and j-th input values to the first and second stages respectively.

As we concerned before, the network compute the weight changes owing to all training patterns, also, add them up, after, update the weights on the basis of the variation of the total weight gather over the full sweep. In the consistent implementation, weight update is committed after each training. So, in the consistent implementation, the weight changes are calculated as following (3) and (4) formulas.

$$\Delta w_{ij} = \rho x_j^p act_i^p (1 - act_i^p)(des_i^p - act_i^p) \quad (3)$$

$$\Delta \vartheta_{jk} = \rho \left[i_k^p x_j^p (1 - x_i^p) \sum_{n-1}^{N} w_{nj} act_n^p (1 - act_n^p)(des_n^p - act_n^p) \right] \quad (4)$$

This approach works very slowly in the series implementation. However, using the SIMD backpropagation approach allows parallelization in the data level. Each network calculates a weight variation vector of the entire network parallel at the same time. After that, the weight vector of the network is updated on the basis of the total weight variation vector. So, owing to parallelization we will achieve more speed. In the next part of our research, we will explain SIMD backpropagation approach.

4.1. Simulation of Backpropagation algorithm in parallel architecture

The main concept of Backpropagation simulation in parallel architecture is the multi layer perceptron. In other words, it is a layered neural network with a different number of hidden layers. Specially, each neuron is associated with each layer neuron of an adjacent layer. For supervised learning we use the Backpropagation algorithm. It is a gradient descent method to determine combination of weights between layers for comparison input and given output values. Feedforward assessment is defined by (5) propagation function.

$$net_j = \sum O_i(t) * w_{i,j} \quad (5)$$

Where, $w_{i,j}$ is represent the connection weights and O_i is the output value of the neuron. Ratio of input, adjacent and output layers is defined by the following equation system, where input, output, hidden are input value, hidden layer's vector, and output vector.

$$\begin{cases} hidden = f(w_1 * input) \\ output = f(w_2 * hidden) \end{cases} \quad (6)$$

Also, w_1 is the weight matrix between the input and hidden layers, and w_2 is the weight matrix between hidden and output layers. The activation function is represented by the sigmoid function (7).

$$y = \frac{1}{1 + e^{-\lambda x}} \quad (7)$$

Also, we use traditional formulas as (4) to learn backpropagation algorithm.

$$\begin{cases} \Delta w_{ij}^1 = \alpha \cdot \varepsilon_i \cdot a_i \cdot (1 - a_i) \cdot h_i \\ \Delta w_{ij}^2 = \alpha \cdot \sum e_m \cdot a_m \cdot (1 - a_m) \cdot w_{mi}^1 \cdot e_j \end{cases} \quad (8)$$

Where, Δw_{ij}^1 and w_{ij}^2 is the weight value of w_1 and w_2 matrixes respectively.

Figure 4 illustrates a schema for single neuron. In the first place, a linear combination of input data will be made.

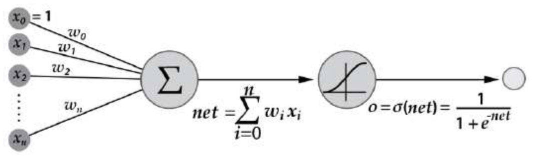
$x_0 = 1$
x_1
x_2
x_n
w_0
w_1
w_2
w_n
$net = \sum_{i=0}^{n} w_i x_i$
$o = \sigma(net) = \frac{1}{1+e^{-net}}$

Figure 4. Schema of single neuron of ANN

Next, we will find the difference between errors of serial and parallel version of backpropagation algorithm. The total mean squared error for a network with n output neurons for a problem with p training patterns is as following:

$$E = \frac{1}{2P}\sum_{p=1}^{P}\sum_{n=1}^{N}(des_n^p - act_n^p)^2 \qquad (9)$$

Next, we consider weight variation between hidden layer and output layer. The received results can be readily generalized to more than one hidden layer in the case of neural network consists of several hidden layers. Even, there is no hidden layer then the hidden layer will be the same as the input layer. Based on the chain rule, we can determine the rate of error change regarding to w_{ij} as the following formula:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial act_i^p} * \frac{\partial act_i^p}{\partial w_{ij}} \qquad (10)$$

Where

$$\frac{\partial E}{\partial act_i^p} = -\frac{1}{P}\sum_{p=1}^{P}(des_i^p - act_i^p)^2 \qquad (11)$$

We consider a sigmoidal activation function as $y = \frac{1}{1+e^{-\lambda x}}$, more precisely it will be in the following form:

$$act_i^p = \frac{1}{a + e^{-\left(\sum_{j=1}^{M} x_j^p w_{ij} + \theta_i\right)}} \qquad (12)$$

Where, M is the number of hidden neurons, x_j^p is the output value of the j-th hidden neuron. And, we get

$$\frac{\partial act_i^p}{\partial w_{ij}} = \frac{x_j^p e^{-\left(\sum_{j=1}^{M} x_j^p w_{ij} + \theta_i\right)}}{\left(1 + e^{-\left(\sum_{j=1}^{M} x_j^p w_{ij} + \theta_i\right)}\right)^2} \qquad (13)$$

$$\frac{\partial act_i^p}{\partial w_{ij}} = x_j^p act_i^p (1 - act_i^p) \qquad (14)$$

Using formulas (10), (11) in equation (14) we get

$$\frac{\partial E}{\partial w_{ij}} = -\frac{1}{P}\sum_{p=1}^{P} x_j^p act_i^p (1 - act_i^p)(des_i^p - of)$$
(15)

Using gradient descent method, the weight variation for w_{ij} we find

$$\Delta w_{ij} = -\frac{\rho}{P}\sum_{p=1}^{P} x_j^p act_i^p (1 - act_i^p)(des_i^p - act_i^p)$$
(16)

Where p is the step size small constant number.

Now, let's consider v_{jk} as the weight connect between k-th input and j-th hidden neurons, and we get

$$(des_n^p - act_n^p)^2 =$$

$$E = \frac{1}{2P}\sum_{p=1}^{P}\sum_{n=1}^{N} \frac{1}{2P}\sum_{p=1}^{P}\sum_{n=1}^{N}\left[des_n^p - \frac{1}{1 + e^{-\left(\sum_{j=1}^{M} x_j^p w_{ij} + \theta_j\right)}}\right]^2$$
(17)

Where,

$$x_j^p = \frac{1}{1+e^{-\left(\sum_{k=1}^{K} x_j^p w_{ij} + \theta_j\right)}} \qquad (18)$$

x_j^p is the output of the j-th hidden neuron for p-th training pattern, K is the number of input neurons, j_k^p is the k-th bit of the p-th training pattern. Using the chain rule one more time, we find

$$\frac{\partial E}{\partial v_{jk}} = \frac{\partial E}{\partial x_j^p}\frac{\partial x_j^p}{\partial v_{jk}} \qquad (19)$$

Using (20):

$$\frac{\partial E}{\partial x_j^p} = \frac{1}{P}\sum_{p=1}^{P}\sum_{n=1}^{N} (des_n^p - act_n^p)\frac{-w_{nj}e^{-\left(\sum_{j=1}^{M} x_j w_{nj} + \theta_n\right)}}{\left(1+e^{-\left(\sum_{j=1}^{M} x_j w_{nj} + \theta_n\right)}\right)^2} = -\frac{1}{P}\sum_{p=1}^{P}\sum_{n=1}^{N} w_{nj} act_n^p (1-act_n^p)(des_n^p - act_n^p)$$

(20)

And

$$\frac{\partial x_j^p}{\partial v_{jk}} = i_k^p x_j^p (1-x_j^p) \qquad (21)$$

we get

$$\frac{\partial E}{\partial v_{jk}} = -\frac{1}{P}\sum_{p=1}^{P} i_k^p x_j^p (1-x_j^p)\sum_{n=1}^{N} w_{nj} act_n^p (1-act_n^p)(des_n^p - act_n^p)$$

(22)

Sharp descent weight change is

$$\Delta v_{jk} = \frac{\rho}{P}\sum_{p=1}^{P} i_k^p x_j^p (1-x_j^p)\sum_{n-1}^{N} w_{nj} act_n^p (1-act_n^p)(des_n^p - act_n^p)$$

(23)

The network computes the weight variation by reason of all the training patterns, and adds them together, after update the weights on the basis on the total weight variation gathered over the whole sweep. In the consistent implementation, the weight renewal is fulfilled after each training pattern. So, using (16) and (23), the weight variation are calculated as the following formulas:

$$\Delta w_{ij} = \rho x_j^p act_i^p (1-act_i^p)(des_i^p - act_i^p)$$

(24)

and

$$\Delta v_{jk} = \rho i_k^p x_j^p (1-x_j^p)\sum_{n=1}^{N} w_{nj} act_n^p (1-act_n^p)(des_n^p - act_n^p)$$

(25)

Figure 5 illustrates descent steps got by moving to the inferior limit of a paraboloid using exact and approximate version of gradient descent algorithm. The SIMD backpropagation method uses the exact algorithm because of it uses parallelism of data level. Each network calculates a vector of weight changes for all weights in the given network on the basis on the given training pattern. After completing the sweep weight variation vectors will be added up and weight vectors are updated on the basis of the weight variation vector. Using the exact method is the result of data level parallelism. In the next part of our research we will explain parallel approach implementation of the problem.

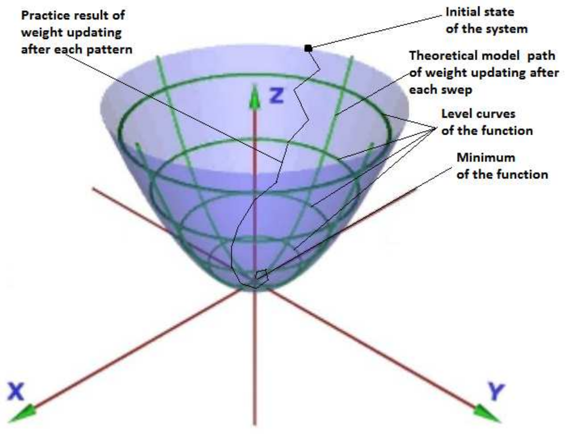
Practice result of weight updating after each pattern
Initial state of the system
Theoretical model path of weight updating after each swep
Level curves of the function
Minimum of the function
Z
X
Y

Figure 5. Descent path to the minimum of a parabolid.

4.2 ANN Parallel Approach

ANN face recognition system consists of two parts as "learning" and "recognition". At the learning part, several images of one person will be given, and the system will be trained with the help of supervisor returning the identification of the given person in the image. At the recognition part, one image will be shown to the camera, and the system will try to find to whom the image belongs.

Figure 6 illustrates the block scheme of the CUDA based on parallel processing face recognition.

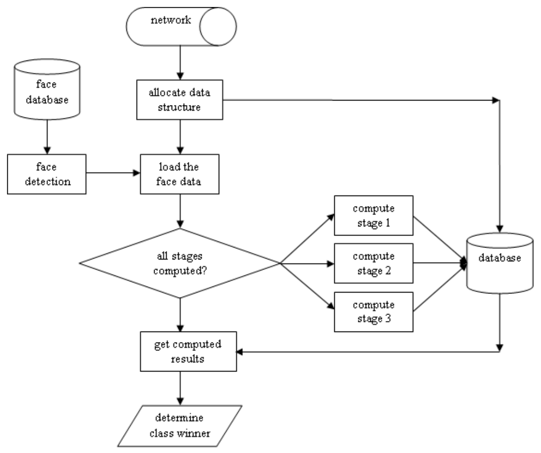

Figure 6. ANN Organization on CUDA

There are two data storages - the one used for storing the weights that are generated during the training stage of the network and the second one used for storing the face images of people who must be recognized. The face recognition algorithm uses a database of 10,000 face images taken from 500 people. In the next part we discuss the results of face recognition and CUDA based parallel processed face recognition.

5. EXPERIMENT RESULTS AND DISCUSSION

A face recognition system generally involves two main stages: face detection and face identification. In the face detection stage, the system searches for any faces. Then, it takes an image of the face. Following this, image processing cleans up the facial image into black-white colors. During the detection process, a common feature for face detection is a set of adjacent rectangles that lie above the eye and the cheek area. The position of these rectangles is defined relative to a detection window that acts like a bounding box to the target object. In our paper, a face can be detected from several foreshortenings. Implemented results are shown in Figure 7.

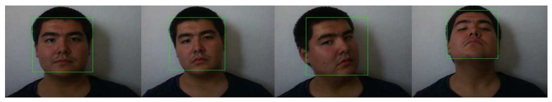

Figure 7. Face detection from several foreshortenings.

After detecting a face, the feature extraction and verification process is performed as a second step. After face recognition, the detected and processed facial image is compared to a database of faces to decide who that person is. Figure 8 illustrates face recognition process. Face recognition occurs by determining id of the recognized person.

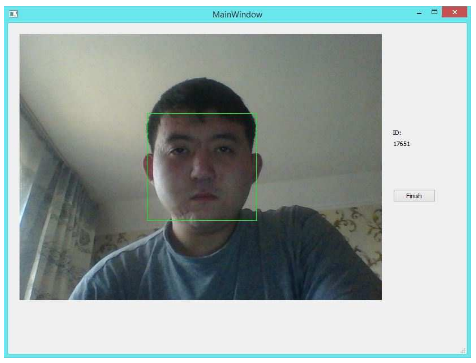

Figure 8. Face recognition process.

5.1. Results

All the benchmarks consist of 2,000 training iterations. Each training iteration include one forward and one backpropagation and the variation of weights. To test the behavior of the system, we used one iteration for a pattern. So, the number of input patterns was 2,000.

To train, we use 10,000 face images from 500 people (there, 1,000 of them are frontal images, the others are the images with some angles), and the system was run in the serial and parallel version. For diversity we take the face images with different emotions such as smile, surprise, sadness, anger, and laugh. Table 2 gives information about characteristics of face images that were used in the experimental part of the research.

Table 2. Characteristics of the images in the database.

Item	Number of images
Face images	10 000
Multiple face images	500
Frontal face images	1 000

5.1.1. Execution Times

In this section, we report and discuss the execution time of operations and the results of the GPU implementing of the proposed SIMD backpropagation algorithm and face compare in face recognition rates in consequently and parallel execution.

5.1.2. Compare sequential and parallel training

Sequential execution time will be solved by next formula:

$$t_{sequential} = (t_1 + t_2) * I + t_3 \qquad (26)$$

Where, t_1, t_2, t_3 are time of forward pass, time of backward pass for training picture, and time for updating the weights respectively. I is number of images.

Parallel execution time will be solved using next formula:

$$t_{parallel} = \{t_{parallel\_1} + t_{parallel\_2}\} * \left(\frac{I}{n}\right) + t_{parallel\_3}$$
(27)

Where, $t_{parallel\_1}$, $t_{parallel\_2}$, $t_{parallel\_3}$ are time of forward pass, time of backward pass for training picture, and time for updating the weights respectively, n is number of nodes.

Speedup ratio will be solved by using formula (8)

$$speedup = \frac{t_{sequential}}{t_{parallel}} \qquad (28)$$

Where, $t_{sequential}$ and $t_{parallel}$ are time spent to forward pass, and backward propagation for training picture, respectively.

Efficiency will find from dividing speed up ration to number of nodes:

$$efficiency = \frac{speedup}{n} \quad (29)$$

5.1.3. Execution time analysis

This subsection compares sequential and parallel execution times (in seconds) and analyses performance results of the algorithm for GPU. Table 3 and Figure 9 report the average execution times for sequential and parallel implementation, and speedup value.

Table 3. Execution times comparing sequential and parallel execution

image size	sequential execution time (second)	parallel execution time (second)	speedup value
30x40	49	54	0.907
60x80	492	185	2.65
90x120	2334	736	3.17
120x160	3109	543	5,72

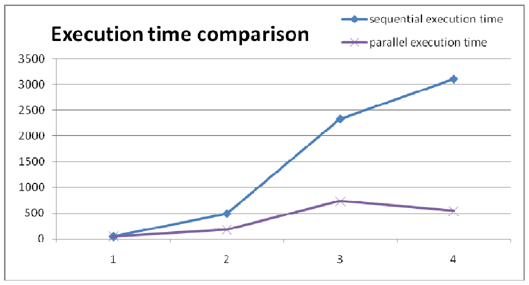

Figure 9. Sequential and parallel execution time comparison.

5.1.4. Speedup analysis

Figure 10 illustrates speedup changes for 30x40 and 120x160 size images when number of input neurons from 8 until 1024. It is clearly seen that, when the size of input neurons grow, parallel training gives great advantage.

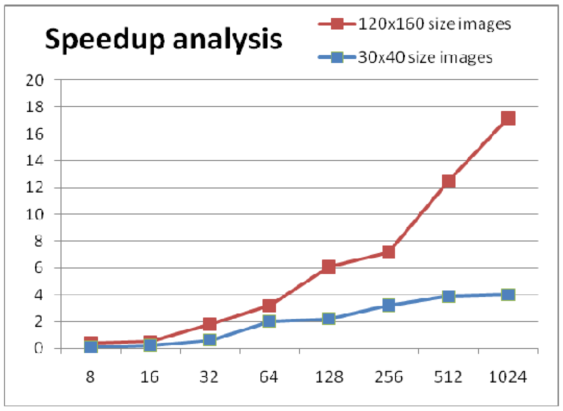

Figure 10. Speedup Analysis depending on hidden neurons.

AUTHOR CONTRIBUTIONS

B.S. Omarov provided ANN training, Parallel Backpropagation algorithms; performed the experiments with parallel and serial face recognition, and explored face detection and recognition algorithms. Also, these results are part of contributions towards his PhD work at Universiti Tenaga Nasional by supervising Azizah Suliman who wrote the outline of the article and guided the direction of our paper, and advised the proof method of the simulation and experimental results. Kaisar Kushibar provided experiments with image processing.

All authors provided substantive comments.

Conflict of interests

The authors declare that there is no conflict of interests regarding the publication of this article.

Acknowledgement

This publication is funded by Malaysian's Ministry of Education (MOE) through the LRGS fund (01201303LRGS).

6. CONCLUSION

We propose the improvement of face recognition using neural network and parallel computing through GPU. The system was described in terms of the mathematical model and neural network algorithms to develop recognition rate and speed up. Therefore, our simulation results confirm the effectiveness of our approach and demonstrate increase in face recognition rate owing to neural network training and fast processing time that is achieved by virtue of GPU CUDA parallelism.

In addition, tackling the problem of fast multiple face recognition is planned by using GPU CUDA and neural network techniques.

Back-propagation is an iterative, gradient search, supervised algorithm which can be viewed as multiplayer non-linear method that can re-code its input space in the hidden layers and thereby solve hard learning problems. The network is trained using ANN technique until a good agreement between predicted gain settings and actual gains is reached.

During last three decades, the assessment of potential of the sustainable eco-friendly alternative sources and refinement in technology has taken place to a stage so that economical and reliable power can be produced. Different renewable sources are available at different geographical locations close to loads, therefore, the latest trend is to have distributed or dispersed power system. Examples of such systems are wind-diesel, wind-diesel-micro-hydro-system with or without multiplicity of generation to meet the load demand. These systems are known as hybrid power systems. To have automatic reactive load voltage control SVC device have been considered. The multi-layer feed-forward ANN toolbox of MATLAB 6.5 with the error back-propagation training method is employed.

REFRENCES:

[1] Ng, C., Savvides, M. and Khosla, P.: Real-time face verification system on a cell-phone using advanced correlation filters, Proc. of 4th IEEE Workshop on Automatic Identifica-tion Advanced Technologies, pp. 57–62. IEEE 2005

[2] Venkataramani, K., Qidwai, S. and Vijayakumar, B.: Face authentication from cell phone camera images with illumination and temporal variations, IEEE Trans. on Systems, Man, and Cybernetics, Part C, vol. 35, pp. 411 – 418. IEEE 2005

[3] Mitchell, Tom: Machine Learning. McGraw Hill 1997

[4] Rumelhart, D., Widrow, B. and Lehr, M.: The basic ideas in neural networks, Communications of the ACM, 37(3) pp. 87-92. ACM 1994

[5] Juan Pablo Balarini, Martín Rodríguez, and Sergio Nesmachnow Centro de Cálculo, Facultad de Ingeniería. Facial Recognition Using Neural Networks over GPGPU. Universidad de la República, Uruguay 2012

[6] U. Seiffert, Artificial neural networks on massively parallel computer hardware, Neurocomputing 57 (2004) 135–150, new Aspects in Neurocomputing: 10th European Symposium on Artificial Neural Networks 2002.

[7] E. Lindholm, J. Nickolls, S. Oberman, J. Montrym, NVIDIA Tesla: A Unified Graphics and Computing Architecture, IEEE Micro 28 (2008) 39–55.

[8] S. Ryoo, C. I. Rodrigues, S. S. Baghsorkhi, S. S. Stone, D. B. Kirk,W.-m.W. Hwu, Optimization principles and application performance evaluation of a multithreaded GPU using CUDA, in: Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming, PPoPP ’08, ACM, New York, NY, USA, 2008, pp. 73–82.

[9] J. Nickolls, I. Buck, M. Garland, K. Skadron, Scalable Parallel Programming with CUDA, Queue - GPU Computing 6 (2008) 40–53.

[10] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, K. Skadron, A performance study of general-purpose applications on graphics processors using CUDA, Journal of Parallel and Distributed Computing 68 (10) (2008) 1370–1380, general-Purpose Processing using Graphics Processing Units. doi:DOI: 10.1016/j.jpdc.2008.05.014.

[11] H. Jang, A. Park, K. Jung, Neural Network Implementation Using CUDA and OpenMP, Digital Image Computing: Techniques and Applications 0 (2008) 155–161.

[12] Sierra-Canto, Xavier, Madera-Ramirez, Francisco, V. Uc-Cetina, Parallel training of a back-propagation neural network using cuda, in: Proceedings of the 2010 Ninth International Conference on Machine Learning and Applications, ICMLA ’10, IEEE Computer Society, Washington, DC, USA, 2010, pp. 307–312. doi:10.1109/ICMLA.2010.52. URL http://dx.doi.org/10.1109/ICMLA.2010.52

[13] S. Lahabar, P. Agrawal, P. J. Narayanan, High performance pattern recognition on gpu, in: National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG’08), 2008, pp. 154–159. URL http://cvit.iiit.ac.in/papers/Sheetal08High.pdf

[14] M. Pethick, M. Liddle, P.Werstein, Z. Huang, Parallelization of a backpropagation neural network on a cluster computer, in: International conference on parallel and distributed computing and systems (PDCS 2003), 2003