PERGAMON

Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Pattern Recognition 37 (2004) 1311–1314

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Rapid and Brief Communication

GPU implementation of neural networks

Kyoung-Su Oh*, Keechul Jung

School of Media, College of Information Science, Soongsil University, 1, SangDo-Dong, DongJak-Gu, Seoul, 156-743, Republic of Korea

Received 6 January 2004; accepted 14 January 2004

Abstract

Graphics processing unit (GPU) is used for a faster artificial neural network. It is used to implement the matrix multiplication of a neural network to enhance the time performance of a text detection system. Preliminary results produced a 20-fold performance enhancement using an ATI RADEON 9700 PRO board. The parallelism of a GPU is fully utilized by accumulating a lot of input feature vectors and weight vectors, then converting the *many* inner-product operations into *one* matrix operation. Further research areas include benchmarking the performance with various hardware and GPU-aware learning algorithms.
© 2004 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.

Keywords: Graphics processing unit(GPU); Neural network(NN); Multi-layer perceptron; Text detection

1. Introduction

Recently graphics hardware has become increasingly competitive as regards speed, programmability, and price. Besides, graphics processing units (GPUs) have already been used to implement many algorithms in various areas, including computational geometry, scientific computation, and image processing, as well as computer graphics [1,2].

In the case of using a neural network (NN) for image processing and pattern recognition, the main problem is the computational complexity in the testing stage, which accounts for most of the processing time. Moreover, NN-based image convolution has to exhaustively scan an input image in order to process an entire image [3]. Although an NN can be simulated using software, many potential NN applications require real-time processing, which means fully parallel specially designed hardware implementations, such as an FPGA-based realization of an NN. However, this is somewhat expensive and involves extra design overheads [4].

Accordingly, the current paper presents a faster NN using common graphics hardware GPU. Although no graphics hardware is dedicated to NN computation, it can still be adapted to many pattern recognition problems with an inexpensive and minimal hardware overhead. The essential operation in an NN is the inner-product between a weight vector and an input vector in each layer. Therefore, to utilize the parallelism of a GPU, lots of input feature vectors and weight vectors are accumulated, then the *many* inner-product operations are converted into *one* matrix operation. As such, 'multiplication' and a 'non-linear threshold function, such as a sigmoid' can be effectively implemented using the *vertex shader* and *pixel shader* in a GPU.

2. Neural network architecture

An artificial neural network, usually referred to as 'neural network', is based on the concept of the workings of the human brain. There are many different types of NN, with the more popular being a multilayer perceptron, learning vector quantization, radial basis function, Hopfield, and Kohonen.

The current study focuses on using a GPU to implement a multilayer perceptron, which is usually fully connected between adjacent layers. The input layer receives the input features of a given application. Although the network

* Corresponding author. Tel.: +82-2-828-7260; Fax: +82-2-822-3622.
E-mail addresses: oks@ssu.ac.kr (K.-S. Oh), kcjung@ssu.ac.kr (K. Jung).

0031-3203/$30.00 © 2004 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2004.01.013

structure can vary as regards the number of layers, number of nodes in each layer, and input mask size, each layer performs the same inner-product operation between the given input vectors and the weight vectors, followed by a non-linear function. Moreover, many inner-product operations can be replaced with a matrix multiplication, which is more appropriate for GPU implementation.

3. GPU processing

Graphics hardware has only been used for rendering within the last few decades, however, its extended capabilities in supporting complex operations have also become useful in non-graphics applications. In particular, the advent of a programmable vertex shader and pixel shader enables flexible functions for general computation. Since GPUs are designed for high-performance rendering where repeated operations are common, they are more effective in utilizing parallelism and more pipelined than general purpose CPUs. Therefore, in areas where repeated operations are common, a GPU can produce a better performance than a CPU.

The mechanism of general computation using a GPU is as follows. The input is transferred to the GPU as textures or vertex values. The computation is then performed by the vertex shader and pixel shader during a number of rendering passes. The vertex shader performs a routine for every vertex that involves computing its position, color, and texture coordinates, while the pixel shader is performed for every pixel covered by polygons and outputs the color of the pixel.

As described above, the inner-product operation for each layer of an NN can be replaced with a matrix multiplication based on accumulating the input vectors and weight vectors. As such, the computation-per-layer can be written as follows:

$$W = \begin{bmatrix} w_{11} & w_{12} & w_{13} & \dots & w_{1N} \\ w_{21} & w_{22} & w_{23} & \dots & w_{2N} \\ \dots & \dots & \dots & \dots & \dots \\ w_{M1} & w_{M2} & w_{M3} & \dots & w_{MN} \end{bmatrix} = \begin{bmatrix} W_1 \\ W_2 \\ \dots \\ W_M \end{bmatrix},$$

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1L} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2L} \\ \dots & \dots & \dots & \dots & \dots \\ x_{N1} & x_{N2} & x_{N3} & \dots & x_{NL} \end{bmatrix} = \begin{bmatrix} X_1 & X_2 & X_3 & \cdots & X_L \end{bmatrix},$$

$$B = \begin{bmatrix} b_1 & b_1 & b_1 & \dots & b_1 \\ b_2 & b_2 & b_2 & \dots & b_2 \\ \dots & \dots & \dots & \dots & \dots \\ b_M & b_M & b_M & \dots & b_M \end{bmatrix}, \tag{1}$$

$$M = W \times X + B = \begin{bmatrix} W_1 \cdot X_1 & W_1 \cdot X_2 & W_1 \cdot X_3 & \dots & W_1 \cdot X_N \\ W_2 \cdot X_1 & W_2 \cdot X_2 & W_2 \cdot X_3 & \dots & W_2 \cdot X_N \\ \dots & \dots & \dots & \dots & \dots \\ W_M \cdot X_1 & W_M \cdot X_2 & W_M \cdot X_3 & \dots & W_M \cdot X_N \end{bmatrix} + \begin{bmatrix} b_1 & b_1 & b_1 & \dots & b_1 \\ b_2 & b_2 & b_2 & \dots & b_2 \\ \dots & \dots & \dots & \dots & \dots \\ b_M & b_M & b_M & \dots & b_M \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & \dots & m_{1L} \\ m_{21} & m_{22} & m_{23} & \dots & m_{2L} \\ m_{31} & m_{32} & m_{33} & \dots & m_{3L} \\ \dots & \dots & \dots & \dots & \dots \\ m_{M1} & m_{M2} & m_{M3} & \dots & m_{ML} \end{bmatrix}, \tag{2}$$

$$R = \text{sigmoid}(M) = \begin{bmatrix} 1+e^{-m_{11}} & 1+e^{-m_{12}} & 1+e^{-m_{13}} & \dots & 1+e^{-m_{1L}} \\ 1+e^{-m_{21}} & 1+e^{-m_{22}} & 1+e^{-m_{23}} & \dots & 1+e^{-m_{2L}} \\ \dots & \dots & \dots & \dots & \dots \\ 1+e^{-m_{M1}} & 1+e^{-m_{M2}} & 1+e^{-m_{M3}} & \dots & 1+e^{-m_{ML}} \end{bmatrix}, \tag{3}$$

where w_{ij} denotes the weight at the connection between the ith node of the output layer and the jth node of the input layer, M is the number of nodes in the output layer, and N is the number of nodes in the input layer. In addition, x_{ij} is the ith feature value of the jth input vector and b_i is the bias term for the ith output node from L input vectors. The final result R_{ij} is the output of the ith output node for the jth input vector.

The above computation comprises of a matrix multiplication followed by a bias factor addition and sigmoid operation. The matrix multiplication is explained first. The method proposed by Moravanszky [1] is used to implement the matrix multiplication. The two matrices are converted into textures, denoted by texture W and texture X, then the matrix multiplication is performed by rendering. A rectangle is rendered to cover the whole screen. The vertex shader outputs the position and texture coordinates for each vertex of the rectangle, where each vertex has two texture coordinates: one for the row of texture W and the other for the column of texture X. For example, the upper left vertex will have the texture coordinates of the first row of texture W and the first column of texture X, while the upper right vertex will have the texture coordinates of the first row of texture W and the last column of texture X, and so on. As a result of

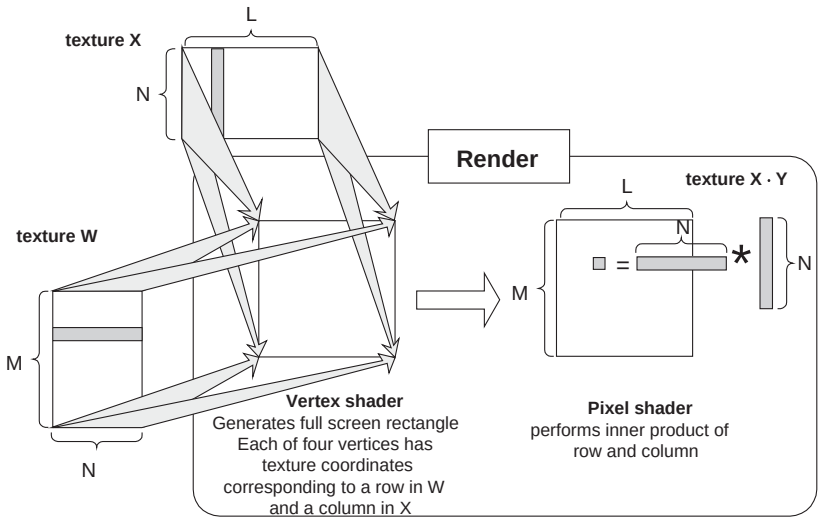

Fig. 1. Overview of matrix multiplication using GPU.

the vertex shader, every pixel (i, j) has texture coordinates corresponding to the ith row of W and the jth column of X. The pixel shader then performs the inner-product between the row of W and the column of X specified by the texture coordinates. Fig. 1 shows an example of matrix multiplication using a GPU. The number of rendering passes required for matrix multiplication depends on the capability of the GPU, including the number of pixel shader operations and number of texture load operations.

The bias term addition and sigmoid operation can be performed in one rendering pass. The bias texture and texture that contains the result of the matrix multiplication, *texture* $W \times X$, are set as the active texture. The vertex shader then outputs a full-screen rectangle as before. Each vertex's texture coordinate for the *texture* $W \times X$ correspond to its position. For example, the upper left vertex has the texture coordinate $(0, 0)$, while the texture coordinate for the upper right vertex is $(1, 0)$. As the bias term is identical for one row, the bias term matrix is one-dimensional and the bias texture coordinates for each vertex correspond to its vertical position. The pixel shader adds two textures and performs a sigmoid operation.

If there is more than one layer in an NN, the above procedure is repeated for each layer. The result of the previous layer is saved in the form of a render target texture, which is then used as an input for the next layer. Note that, even though an NN may have multiple layers, the GPU can perform all the operations after texture creation.

4. Application to pattern recognition

Recently, researchers have attempted text-based retrieval of image and video data using several image processing techniques [3]. As such, an automatic text detection algorithm for image data and video documents is important as a preprocessing stage for optical character recognition, and an NN-based text detection method has several advantages over other methods [3].

Therefore, this paper briefly describes such a text detection method, and readers are referred to the author's previous publication for more details [3]. In the proposed method, an NN is used to classify the pixels of input images, whereby the feature extraction and pattern recognition stage are integrated in the neural network. The NN then examines local regions looking for text pixels that may be contained in a text region. Therefore, an M × M pixel region in the image is received as the input and a classified image is generated as the output. After the pattern passes the network, the value of the output node is compared with a threshold value and the class of each pixel determined, resulting in a classified image. GPU-based pipelining processing is used to reduce the processing time, and the GPU's performance is maximized by accumulating a large number of input vectors [1] to create a two-dimensional texture. The input layer then receives the grey values for the pixels at predefined positions inside an M × M window over the input image. Experiments were conducted using an 11 × 11 input window size, with the number of nodes in each hidden layer set at 30. As a result, the processing time for pixel classification was significantly reduced using a GPU. Fig. 2(b) shows the pixel classification result for the left input image, where a black pixel denotes a text pixel. The classification using a GPU produced almost the same result as without a GPU.

[1] It is dependent on the GPU configuration. The maximum texture size of an ATI RADEON 9700 PRO board is 2048.

march 2002

Who was your favorite teacher?

The one who stayed late after class to help you solve a difficult problem? The one who inspired you to follow your dream? The one who listened to your problems? The role of a teacher extends far beyond the classroom walls preparing today's students for tomorrow's challenges.

Teacher, Teacher, Can you teach me?

by Jessica Calleja

Education is vital to a country's long-term success. Many teachers will be retiring in the next ten years creating a shortage of qualified teachers. This has led many districts and school boards to lobby for focused training, increased benefits and more support. Now is the time to seriously consider this ageless profession.

Not Enough Teachers To Go Around

Experts estimate that 150,000 to 250,000 teachers will be hired in the next five years to meet the demand in the nation's elementary and secondary schools. This means over 2.2 million Americans will be entrusted with the task of educating future generations. The demand for teachers is nation wide with needs varying from region to region. Specific subject and geographic areas with immediate shortages include: Mathematics, Sciences, Bilingual Education/English as a Second Language, Special Education, Urban/Rural Schools, Ethnic Teachers and Male Teachers.

Marilyn Kuehlem, project co-ordinator for the Texas Teacher Recruitment Retention Project with the Texas Education Agency is working towards attracting graduates to this profession. She outlines several reasons for the predicted shortage that are not only common in Texas but throughout the entire country.

"It is pretty well documented in Texas that we need certified teachers. This year 25 per cent of those hired were not certified in the area in which they were teaching.

Those going into the university system at this point are not keeping up with the predicted demand for students and in Texas student enrollment has been growing at a pace of about 20 per cent over the last 10 years," says Kuehlem. "We have information from the Teacher Retirement System that shows in Texas more that 116,000 teachers will be eligible to retire by the year 2010. That's not too far away. What we are trying to do with the Teacher Recruiting Retention Campaign is recruit more into the profession." Many state school districts are following suit recruiting today to fill gaps that are predicted in the near future.

The Latest Trends

Many districts are now creating innovative strategies to compete for a shrinking pool of qualified candidates. Examples of incentives being offered to attract and retain teachers are:

Signing Bonuses

- Massachusetts created a $20,000 signing bonus program for college graduates and mid-career professionals willing to teach in the state for four years.
- Several cities in Texas have offered signing bonuses of $1,500 - $2,000 for teachers who fulfill specific hiring needs, such as secondary math and science.
- The New York Board of Regents has recommended a $10,000 signing bonus for teachers who agree to work in high-need schools for three years.

Help with Real Estate Costs

- Baltimore teachers received up to $5,000 toward closing costs on a home in the city.

Higher Starting Salaries

- Nevada's average starting salary exceeded all adjacent states and the national average for 1996-97.
- New Jersey raised average salaries to compete more effectively with careers outside of teaching, and now ranks as the state with the highest average salary in the nation.

Incentives for Veteran Teachers

- Montgomery County, CA offers

continued on page 8

6

(a) (b)

Fig. 2. Experimental Results: (a) test image, (b) result of MLP with GPU.

Table 1
Processing times per elementary operations

	Texture creation	Matrix multiplication	Sigmoid
GPU	0.469000	0.030000	0.031000
CPU		11.743	

As shown in Table 1, we get a 20-fold performance enhancement using an ATI RADEON 9700 PRO board compared to CPU-only processing.

Acknowledgements

This work was supported by the Soongsil University Research Fund.

References

[1] A. Moravanszky, Linear algebra on the GPU, in: W.F. Engel (Ed.), Shader X 2, Wordware Publishing, Texas, 2003.

[2] D. Manocha, Interactive geometric & scientific computations using graphics hardware, SIGGRAPH 2003 Tutorial Course #11.

[3] K. Jung, Neural network-based text location in color images, Pattern Recog. Lett. 22 (14) (2001) 1503–1515.

[4] J. Zhu, P. Sutton, FPGA implementation of neural networks —a survey of a decade of progress, Proceedings of the 13th International Conference on Field Programmable Logic and Applications (FPL 2003), Lisbon, 2003, pp. 1062–1066.