ISSN:0975-9646

Fast Efficient Artificial Neural Network for Handwritten Digit Recognition

Viragkumar N. Jagtap[1] , Shailendra K. Mishra[2]

[1]*M.E Student, Parul Institute of Technology, Vadodara.*

[2]*Department of CSE,Parul Institute of Technology, Vadodara.*

Abstract: -Handwriting recognition is having high demand in commercial & academics. In recent years lots of good work has been done on hand written digit recognition to improve accuracy. Handwritten digit recognition system needs larger dataset and long training time to improve accuracy & reduce error rate. Training of Neural Networks for large data sets is very time consuming task on CPU. Hence, in this paper we presented fast efficient artificial neural network for handwritten digit recognition on GPU to reduce training time. Standard back propagation (BP) learning algorithm with multilayer perceptron (MLP) classification is chosen for this task & implemented on GPU for parallel training. This paper focused on specific parallelization environment Compute Unified Device Architecture (CUDA) on a GPU hence effectively speedup training & reduce training time.

Keywords: Artificial Neural Networks (ANN), Multilayer Perceptron (MLP), Parallel Training, Back Propagation (BP) Graphics Processing Unit GPU, CUDA

1. INTRODUCTION

To recognize handwritten digits like humans or near to that is very challenging task. Because of long training time of learning algorithms it is difficult to bring it to commercial application. To bridge this gap combination of software and hardware is required that works on parallelization [1],[2]. For high accuracy large amount of data is required to train neural nets. But there is a greater need to explore the capabilities of advance hardware and software technology by exploring the parallelization capabilities of graphic processing units (GPUs). The CUDA is a parallel computing platform and programming environment invented by NVIDIA [4]. In this paper we explore the features of CUDA on GPU for the parallelized simulation of a neural network based handwritten digit recognition. The paper is organised as follows. Section 2 presents the related literature work, state of the art and the recent research trends. Section 3 describes the Parallel Training Approach and implementation details. Section 4 describes experimental setup, Neural Network Architecture & dataset. Section 5 describes the performance & observations. Finally, section 6 gives the conclusion and future work.

2. LITERATURE REVIEW

Lots of research papers are written on handwritten digit recognition. An efficient Hindi Digit Recognition System drawn by the mouse and developed using Multilayer Perceptron Neural Network (MLP) with Backpropagation [5]. The proposed system has been trained on samples of 800 images and tested on samples of 300 images written by different users selected from different ages. An experimental result shows high accuracy of about 91% on the testing samples. In [6] author presented an hybrid approach with MLP for Gujarati hand written numerals And achieved accuracy of about 92% on the testing samples. A novel approach using SVM with MLP for English hand written numerals is having highest accuracy of about 97% on the testing samples [7]. A novel approach using ANN with Hu moments for English hand written numerals with poor accuracy on the testing samples [8]. ANN with nprTool for English hand written numerals And achieved highest accuracy of about 98% on the testing samples [9]. Processing steps for handwritten digit recognition system is as follows.

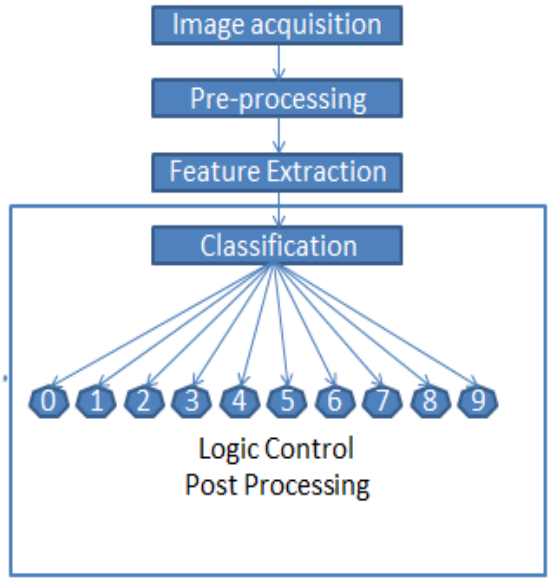

Fig.1. Processing steps for handwritten digit recognition system.

Advance technology we have of powerful graphic processing units (GPU) in our desktop, laptops and servers at low cost. Applying parallelization techniques for neural network simulation became a promising research field by using modern hardware.

Table 1: Focus on training time from previous work

Title	*Approach*	*Accuracy*	*Data Size*	*Focus Training Time*
"An Efficient Neural Network For Recognizing Gestural Hindi Digits" AJAS 2013 [5]	Back Propagation with MLP	91%	small	No
"Recognition Of Gujarati Numerals Using Hybrid Approach And Neural Networks " IJCA(2013)[6]	Hybrid with MLP	92%	small	No
" A Novel Approach to Recognize the off-line Handwritten Numerals using MLP and SVM Classifiers" IJCSET (2013)[7]	SVM with MLP	97%	Very small	No
"An Artificial Neural Network Model For Handwritten Digits Recognition" RRTIEP (2013)[8]	ANN with Hu moments	Poor	small	No
"Handwritten Isolated Digit Recognition Using Artificial Neural Networks" IJCSAIT(2013)[9]	ANN with nprtool	98%	Very small	No

3. PROPOSED SYSTEM

3.1 Overview of Proposed Work

We proposed new fresh approach to existing work for improvements of learning algorithm and novel feature extraction method for Handwritten Digit Recognition. We focus to implementing method and architecture uses GPUs for parallel processing to speed up learning process

Our study forces us to improve following parameters to be considered to improve the system.

1) Decrease training time without affecting accuracy
2) Implementing method and architecture uses GPU for parallel processing to speed up learning process.

3.2 Flow of Proposed Work

1) Input the digit: The user draws a digit inside the special window using the mouse and then it is saved on a file as .bmp/.jpg image
2) Pre-processing: The goal of pre-processing is to simplify the digit recognition problem without throwing away any important information to be more concise representation for feature extraction stage. This operation involves converting the gray image into a binary image, perform skeletonization, invert the image, and resize the binary image
3) Feature extraction: In this work novel Features Extraction Method is proposed. Features are a set of values of a given digit that are used to distinguish the digit from each other. The feature extraction phase calculates these values in order to produce a set of measurements, called the feature vector, for each object
4) Learning Phase: In this work new parallel learning method is approached.
5) Digit recognition: The recognition step is based on the use of neural networks, or in more, it's based on MLPs. This step realizes a set of discriminated functions that associate a score to each possible class. These scores may be regarded as being representative of the probability of each class, to be one of the digits presented to the system
6) Display digit: Displaying the output of a digit.

3.3 Back-Propagation Algorithm [5][12][14]

Back-propagation algorithm consists of the following steps:

1) Initialize input layer including an input for bias. I_i, W_i, T_i, Y_i
   Where, I_i = input neurons, W_i = random weights, T_i= target values, Yi=output at each neuron
2) Propagate activity forward through input layer to output layer
   $I \Rightarrow H \Rightarrow O$
3) Calculate output at each neurons at each layer
   $O_i = \sum I_i * W_i$
4) Apply activation function to neurons and collect final output at each neurons
   $Y_i = 1/1+e^{-(Oi)}$
5) Calculate the error in the output layer
   $Error = 1/2 (Y_i-T_i)^2$
6) Back propagate the error through layer
   $dE/dW_t = d(Error)/dW_t$
7) Update the weights
   $\Delta W_t = - \varepsilon (dE/dW_t) + \alpha(\Delta W_{t-1})$
   Where,
   ε = learning rate &
   α = momentum

Algorithm 1: Back propagation on CPU

Forward Propagation

1. $O_i = I_i W_1$
2. $O_i = 1/1+e^{-\sum Ii * Wi}$
3. $Y_i = O_i W_2$
4. $Y_i = 1/1+e^{-Yi}$

Backward Propagation

5. $Error = 1/2 (Y_i-T_i)^2$
6. $\Delta E = d(Error) / dW_t$
7. $\Delta W_t = - \varepsilon (dE / dW_t) + \alpha(\Delta W_{t-1})$
8. $W_j <= W_j + \Delta W_t$

3.4 Proposed Parallel Learning Method

We apply parallelization approach motivated by our problem domain and the available hardware resources:

1. The network node parallelization is deployed on GPU using CUDA.

2. In this only one copy of the network is instantiated, which resides on GPU.
3. Each thread on the GPU behaves like a neuron and executes independently.
4. To speed up the implementation the training weights and input data are stored in one dimensional array aligned with the host and the device memory for looping in GPU.
5. BPMLP is trained by a supervised learning mode. After a feed-forward operation, the output value is compared with the target value and classified to 10 categories, to check whether the BPMLP has correctly classified the hand written digits.
6. GPU implementations make use of the existing timer function clock gettime () to record the system time and calculate the overall execution time.

3.4.1 Feed-Forward Phase

The GPU program aims for a fully parallelization of the feed-forward operation using a huge number of threads to avoid any looping. This means every GPU-thread is therefore responsible for the computation of only a single weight value.

Thus 28*28=784 threads were launched for a BPNN with 784 inputs and 533 hidden neurons in the feed-forward phase from the input to the hidden layer.

Each thread has read access to a single input neuron, read access to a single weight value and write access to a hidden neuron, resulting in (n + 1) *h write operations in the hidden layer.

The number of threads was reduced to 533 threads for the above mentioned use-case.

Each thread has read access to all input layer neurons, read access to a column of weights concerning a specific hidden neuron (edges from each input neuron to this particular hidden neuron) and write access to the same single hidden neuron.

Therefore the write accesses to the hidden layer are reduced to h, each thread having only to perform a single write operation.

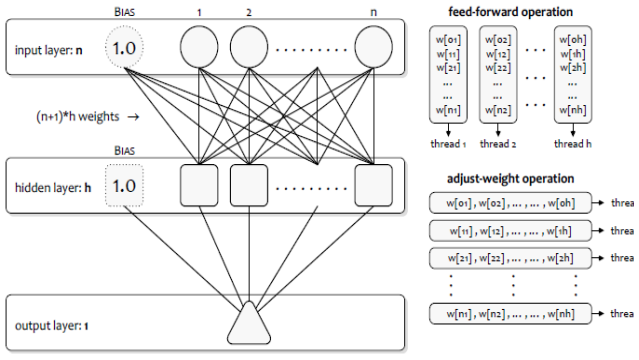

Fig.2. Parallel Training Approach

3.4.2 Adjust-Weight Phase

In the GPU implementation two versions of the adjust-weight operation may implemented and analysed a row-oriented version, where all weights of an input node (layer 1) are handled by a thread, and a column-oriented version, where all weights of a hidden node (layer 2) are handled by a thread. The row-oriented version results in a lower execution time than the column-oriented version due to the given physical layout of the weight matrix and its values that are allocated row-wise in memory. The row-oriented version reaches a better spatial locality concerning the write operations and was therefore chosen for the experiments.

Algorithm 2: Parallel Learning Method (Back propagation) on GPU

1) Clean Host & Devices
2) Initialize Host(topology, learningRate, momentum, minWeight, maxWeight)
3) InitializeDevice(Layer, Neuron, Connection)
   // *Forward Propagation*
4) Call CUDA Kernel Feed Forward
   FeedForward ();
   // *Backward Propagation*
5) Call CUDA Kernel Calculate gradients
   //Calculate gradients
   CalcHiddenLayerGradients ();
   CalcOutputLayerGradients ();
   //weight updation
   Call CUDA Kernel
   UpdateConnectionWeights ();
6) Get results from Device to host
   getResults ()
   {cudaMemcpy (cudaMemcpyDeviceToHost);}
7) Calculate average error using
   CalcOutputError();
8) Persist network model file
   Persist(filename)
   {//After training network will Save network model file "abc.net" }
9) Load Network Model for recognition
   Load (fileName)*{//Load network file for recognition file}*

4. Experimental Setup

4.1 Architecture

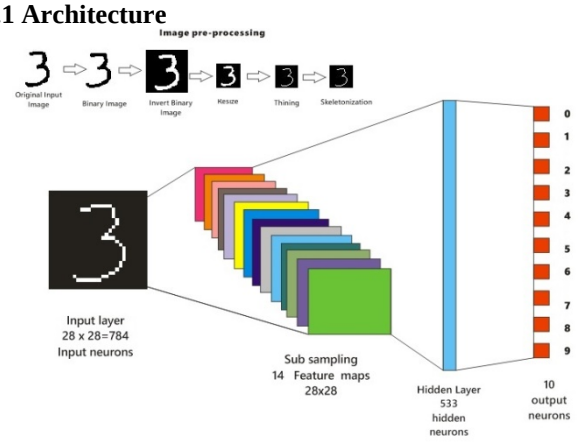

Fig.3 VNet Architecture (Topology)

Our study shows that this ongoing hardware progress may be more important than advance in algorithms and software (although the future will belong to methods combining the best of both worlds). Current graphics cards (GPUs) are

already more than 40 times faster than standard microprocessors [4].

Table.5 Training parameters

Topology	784,533,10
Weight range	-1,1
Learning rate	0.1
Momentum	0.9
Error Threshold	0.5

We train MLP with one input, one hidden & one output layer and varying numbers of hidden units. Mostly but not always the number of hidden units per layer decreases towards the output layer. Weights are initialized with a uniform random distribution in [-1, 1]. The initially random weights of the BMLP are iteratively trained to minimize the classification error on a set of labelled training images; generalization performance is then tested on a separate set of test images.

4.2 Dataset Description

We chose MNIST initially for training BPMLP.MNIST consists of two datasets, one for training (60,000 images) and one for testing (10,000 images). Pixel intensities of the original gray scale images range from 0 (background) to 255 (max foreground intensity). 28 x 28 = 784 pixels per image are fed into the NN input layer. After working on MNIST we started building our own dataset HDDIL.

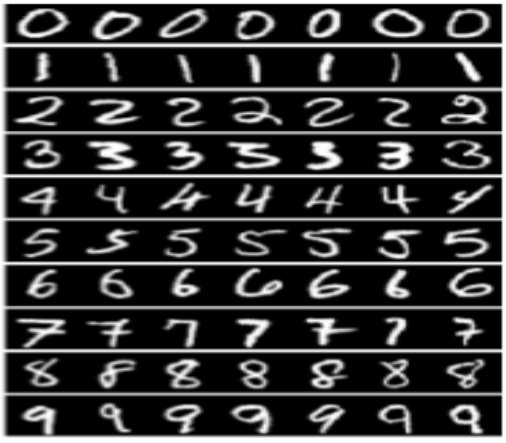

Fig.4. 28 x 28 bitmap images for handwritten digits from MNIST

5. Performance Observations

5.1 Computation Environment

For experiment we use the following hardware and software environment: Intel CORE i5 machine with 4GB memory at 800MHz. The multithreaded GPU program was compiled by CUDA 5.5 and runs on nvidia GeForce GT 630 graphics card with 1GB memory & 96 CUDA Cores.

5.2 Performance Analysis

5.2.1 Speedup factor

For our simulation run the GPU programs used the same BPMLP configuration (learning rate, momentum, number of neurons, initialized weights and number of epochs). Therefore we can directly compare the execution times of the different runs. For all runs we set the learning rate (0.1) and the momentum (0.9) and vary only the number of epochs. As compared to serial execution on CPU for a BPMLP with 533 hidden neurons the GPU program (speedup = *SP*) will outperforms the CPU program. Performance factor = (T_{CPUs} / T_{GPUs}) = 2617/ 1070 = 2.445794392523364

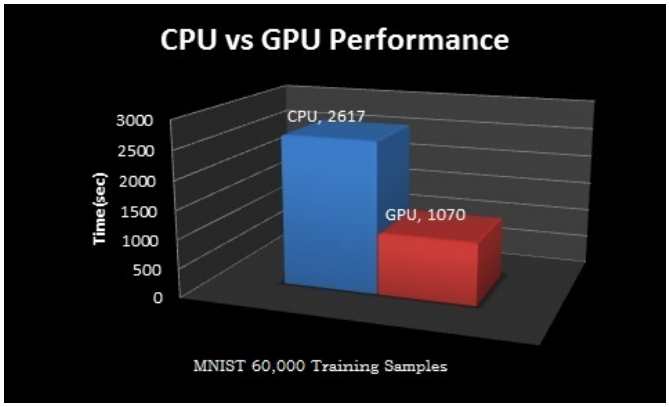

Fig.5 Comparison with existing system Speedup Analysis for 60000 training samples

5.2.2 Accuracy

From our experiment we obtain average 97.7% recognition accuracy on actual data and 98% on test data. We successfully improved accuracy then current system. Fig.36. shows recognition rate of proposed system for 0 to 9 digits & Fig.37shows comparison of accuracy between proposed vs existing system for handwritten digits.

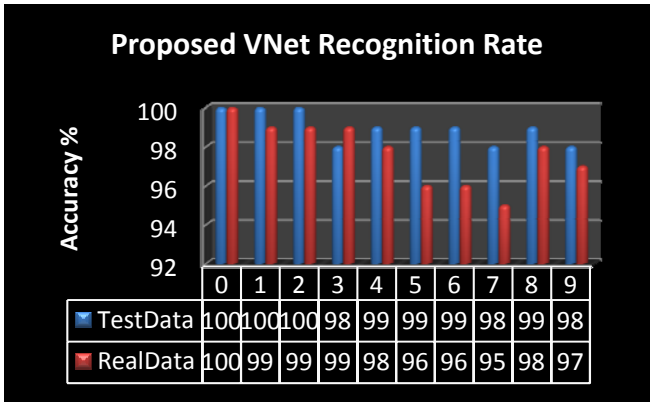

Fig.6 Recognition rate of proposed system for 0 to 9 digits

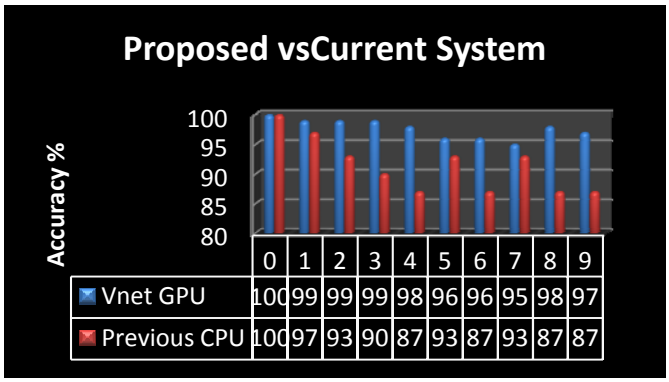

Fig.7 Comparison with existing system recognition rate for handwritten digits

6. CONCLUSION AND FUTURE SCOPE

In this paper we presented fast efficient artificial neural network for handwritten digit recognition on GPU to reduce training time with PTM (Parallel Training Method). We derived back propagation algorithm on GPU based parallelization should be preferred generally with compared to CPU based program. But still, for back propagation with small input data and few hidden neurons CPU based execution is better. But, if the input dataset is larger than GPU based parallelization is suitable to reduce training time.

We expecting to explore the capabilities of GPU and multicores in cloud environments and offering them as services, where users can query and select them, depending on respective service level agreements and the system is providing high performance by automatic parallelization.

REFERENCES

[1] D. Nguyen, B. Widrow, Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights, in: Proceedings of the international joint conference on neural networks, Vol. 3, Washington, 1990, pp. 21–26.

[2] P.Wu, S.-C. Fang, H. Nuttle, Curved search algorithm for neural network learning, in: Neural Networks, 1999. IJCNN '99. International Joint Conference on, Vol. 3, 1999, pp. 1733 –1736 vol.3. doi:10.1109/IJCNN.1999.832638.

[3] CUDA Specifications and Documentation.URL http://docs.nvidia.com/cuda/index.html

[4] Nidal Fawzi Shilbayeh, Mohammad Mahmmoud Alwakeel And Maisa Mohy Naser, "An Efficient Neural Network For Recognizing Gestural Hindi Digits" American Journal Of Applied Sciences 10 (9): 938-951, ISSN: 1546-9239 ©2013

[5] Baheti M. J. ,Kale K. V. , "Recognition Of Gujarati Numerals Using Hybrid Approach And Neural Networks " International Journal Of Computer Applications (0975 – 8887) International Conference On Recent Trends In Engineering & Technology - 2013

[6] Mamta Garg Et Al. " A Novel Approach to Recognize the off-line Handwritten Numerals using MLP and SVM Classifiers" International Journal Of Computer Science & Engineering Technology (IJCSET) ISSN : 2229-3345 Vol. 4 No. 07 Jul 2013

[7] Snezana ZEKOVICH Milan TUBA , "An Artificial Neural Network Model For Handwritten Digits Recognition" Recent Researches In Telecommunications, Informatics, Electronics And Signal Processing ISBN: 978-960-474-330-8 2013

[8] K. Siva Kumar, T.Anusha, B .Ramesh, "Handwritten Isolated Digit Recognition Using Artificial Neural Networks" International Journal Of Computer Science Applications & Information Technologies Vol.1, No.1 (2013)

[9] Dan C. Ciresan, Jonathan Masci, Luca M. Gambardella, Jurgen Schmidhuber "Handwritten Digit Recognition with a Committee of Deep Neural Nets on GPUs" Technical Report No IDSIA-03-11, March 2011.

[10] S N Sivanandam, S Sumathi, S N Deepa "Introduction to Neural Networks Using MATLAB6.0" Mc Graw Hill Education 2013

[11] Simon, Hykin "Neural Networks and Learning Machines", PHP 2013

[12] Han, Kamber, Pei "Data mining Concepts & Techniques", Morgan Kaufman MIT press.