IJCRT.ORG ISSN : 2320-2882
INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)
An International Open Access, Peer-reviewed, Refereed Journal

Detection of Cacography Notation using Rapid and Proficient Artificial Neural Network

Prof. Uttam Kumar Jena[1], Ms Prajna Priyadarshini Mohanty[2], Prof. Swarupa Paattnaik[3]
Department of Computer Science and Engineering
Raajdhani Engineering College, Bhubaneswar

Abstract: -Cacography Detection is having appeal in business and scholastics. As of late loads of good work has been done on transcribed digit acknowledgment to improve precision. Manually written digit acknowledgment framework needs bigger dataset and long preparing time to improve exactness and diminish mistake rate. Preparing of Neural Organizations for huge informational collections is tedious undertaking on central processor. Henceforth, in this paper we introduced quick effective counterfeit neural organization for manually written digit acknowledgment on GPU to decrease preparing time. Standard back proliferation (BP) learning calculation with multi-facet perceptron (MLP) characterization is picked for this undertaking and executed on GPU for equal preparing. This paper zeroed in on explicit parallelization climate Process Bound together Gadget Engineering (CUDA) on a GPU consequently adequately speedup preparing and diminish preparing time.

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

1. Introduction

To perceive written by hand digits like people or close to that is testing task. As a result of long preparing season of learning calculations it is hard to carry it to business application. To overcome this issue mix of programming and equipment is necessitated that deals with parallelization [1],[2]. For high precision huge measure of information is needed to prepare neural nets. Yet, there is a more noteworthy need to investigate the abilities of advance equipment and programming innovation by investigating the parallelization capacities of realistic handling units (GPUs). The CUDA is an equal figuring stage and programming climate created by NVIDIA [4]. In this paper we investigate the highlights of CUDA on GPU for the parallelized reproduction of a neural organization based manually written digit acknowledgment. The paper is coordinated as follows. Segment 2 presents the connected writing work, best in class and the new exploration patterns. Area 3 portrays the Equal Preparing Approach and execution subtleties. Area 4 portrays exploratory arrangement, Neural Organization Design and dataset. Segment 5 depicts the exhibition and perceptions. At last, area 6 gives the end and future work.

2. Literature Review

Perceptron Neural Network with Back propagation [5]. The proposed framework has been prepared on examples of 800 pictures and tried on examples of 300 pictures composed by various clients chose from various ages. A trial result shows high exactness of about 91% on the testing tests. In [6] writer gave a half and half methodology MLP for Gujarati transcribed numerals And accomplished precision of about 92% on the testing tests. An epic methodology utilizing SVM with MLP for English manually written numerals is having most elevated exactness of about 97% on the testing tests [7]. A tale approach utilizing ANN with Hu minutes for English transcribed numerals with helpless exactness on the testing tests [8]. ANN with nprTool for English written by hand numerals and accomplished most noteworthy exactness of about 98% on the testing tests [9]. Handling ventures for written by hand digit acknowledgment framework is as per the following.

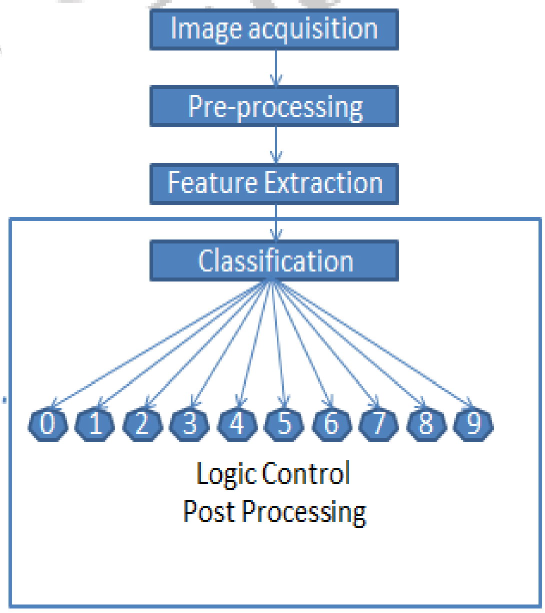

Fig.1. Processing steps for handwritten digits recognition system

Heaps of examination papers are composed on transcribed digit acknowledgment. A proficient Hindi Digit

Acknowledgment Framework drawn by the mouse and created utilizing Multilayer

Advance innovation we have of amazing realistic handling units (GPU) in our work area, workstations and workers with ease. Applying parallelization procedures for neural organization recreation turned into a promising exploration field by utilizing current equipment.

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
"Handwritten Isolated Digit Recognition Using Artificial Neural Networks" IJCSAIT(2013)[9]	ANN with nprtool	98%	Very small	No

3. PLANNED SYSTEM

Overview of Proposed Work

We planned new bright approach to obtainable work for improvements of learning algorithm and pristine feature extraction method for Cacography notation detection. We focus to applying method and architecture uses GPUs for parallel allowance to speed up learning development

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

1) Diminish training time with no affecting correctness
2) Executing method and architecture uses GPU for parallel dispensation to speed up learning procedure.

Flow of Proposed Work

1) Input the notation: 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 objective of pre-processing is to make simpler the notation detection problem without chucking away any important information to be more briefing representation for feature removal stage. This operation engages converting the gray picture into a binary image, perform skeletonization, invert the image, and resize the binary picture.
3) Feature extraction: In this work new Features Extraction Method is planned. Features are a set of values of a given digit that are used to distinguish the digit from each other. The feature extraction phase computes these values in order to create a set of measurements, called the feature vector, for each object.
4) Learning Phase: In this effort new parallel learning method is move toward.
5) Notation Detection: The detection step is based on the use of neural networks, or in more, it's based on MLPs. This step understands a set of discriminated functions that connect a score to each possible class. These scores may be observed as being representative of the probability of each class, to be one of the notations presented to the system.
6) Display notation: Displaying the output of a notation.

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

Back-propagation algorithm consists of the subsequent 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, Y_i=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

   $$Y_i = 1/1+e^{-(O_i)}$$
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

Forward Propagation

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

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. $W_j \Leftarrow W_j + \Delta W_t$

Proposed Parallel Learning Method

We affect parallelization move toward provoked by our problem area and the obtainable hardware assets:

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

2. In this only one copy of the network is instantiated, which exists in on GPU.
3. Every thread on the GPU performs like a neuron and executes separately.
4. To speed up the execution the training weights and input data are accumulated in one structural array united 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 evaluated with the target value and classified to 10 categories, to check whether the BPMLP has properly classified the notations.
6. Graphic Processing Unit applications make use of the existing timer function clock get time () to record the system time and compute the overall finishing time.

Feed-Forward Phase-:

The Graphic Processing Unit program focuses on a completely parallelization of the feed-forward activity utilizing an enormous number of strings to abstain from any circling. This implies each GPU-string is along these lines answerable for the calculation of just solitary weight esteem.

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 string has perused admittance to a solitary information neuron, read admittance to a solitary weight esteem and compose admittance to a secret neuron, bringing about (n + 1) *h compose tasks in the secret layer.

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

Each string has perused admittance to all information layer neurons, read admittance to a segment of loads concerning a particular secret neuron (edges from each information neuron to this specific secret neuron) and compose admittance to a similar single secret neuron.

Consequently the write admissions to the hidden layer are reducedto h, each thread having only to perform a single write operation.

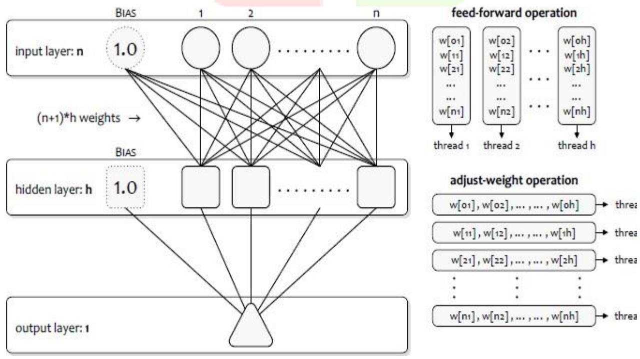

Fig.2. Parallel Training Approach

Quadrate-Weight Phase

In the Graphic Processing Unit application, two versions of the adjust-weight operation may implemented and analyzed a row- oriented version, where all loads of an input node (layer 1) are handled by a thread, and a column-oriented description, where all loads of a secret hub (layer 2) are dealt with by a string. The line arranged adaptation brings about a lower execution time than the segment situated form because of the given actual design of the weight framework and its qualities that are apportioned column shrewd in memory. The line situated rendition arrives at a superior spatial region concerning the compose activities and was subsequently picked for the examinations.

Algorithm 2: Parallel Learning Method (Back propagation)

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

Architecture

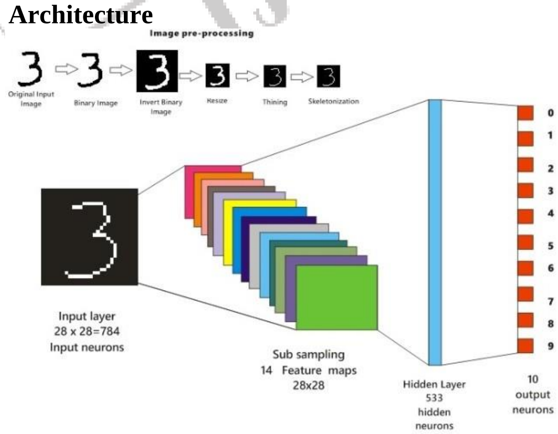

Fig.3 VNet Architecture (Topology)

This work shows that this ongoing hardware growth may be more significant than advance in algorithms and software (although the future will belong to methods merge the best of both worlds). Current graphics cards are previously more than 40 times faster than typical 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 info, one covered up and one yield layer and changing quantities of covered up units. Generally however not generally the quantity of covered up units per layer diminishes towards the yield layer. Loads are instated with a uniform irregular circulation in [-1, 1]. The at first arbitrary loads of the BMLP are iteratively prepared to limit the arrangement mistake on a bunch of marked preparing pictures; speculation execution is then tried on a different arrangement of test pictures.

Dataset Explanation

We picked MNIST at first for preparing BPMLP. **MNIST** comprises of two datasets, one for preparing and one for testing. Pixel powers of the first dim scale pictures range from 0 (foundation) to 255 (max frontal area force). 28 x 28 = 784 pixels for every picture are taken care of into the NN input layer. Subsequent to chipping away at MNIST we began building our own dataset **HDDIL**.

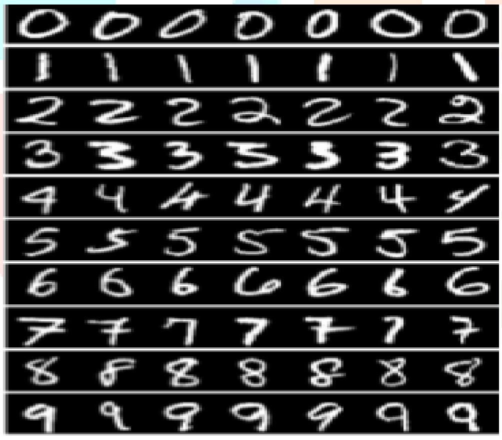

Fig.4. 28 x 28 bitmap images for handwritten notations from MNIST

5. Performance Annotations

Computation Environment

For try we utilize the accompanying equipment and programming climate: Intel Center i5 machine with 4GB memory at 800MHz. The multithreaded GPU program was ordered by CUDA 5.5 and runs on nvidia GeForce GT 630 illustrations card with 1GB memory and 96 CUDA Centers.

Performance study

For our reproduction run the GPU programs utilized a similar BPMLP design (learning rate, energy, number of neurons, introduced loads and number of ages). Subsequently we can straightforwardly think about the execution seasons of the dissimilar runs. For all runs we put the learning rate (0.1) and the momentum (0.9) and vary only the number of epochs. As evaluated to serial execution on CPU for a BPMLP with 533 hidden neurons the GPU program (speedup = *SP*) will depicts the CPU program. Performance factor = (T_{CPUs} / T_{GPUs}) = 2617/ 1070 = 2.445794392523364

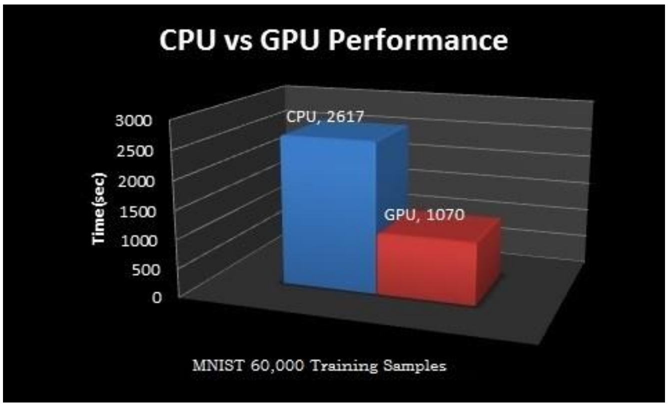

Fig.5 Comparison with existing system Speedup Analysis for 60000 training samples

Accurateness

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.6. shows recognition rate of proposed system for 0 to9 digits & Fig.7shows comparison of accuracy between proposed vs existing system for handwritten digits.

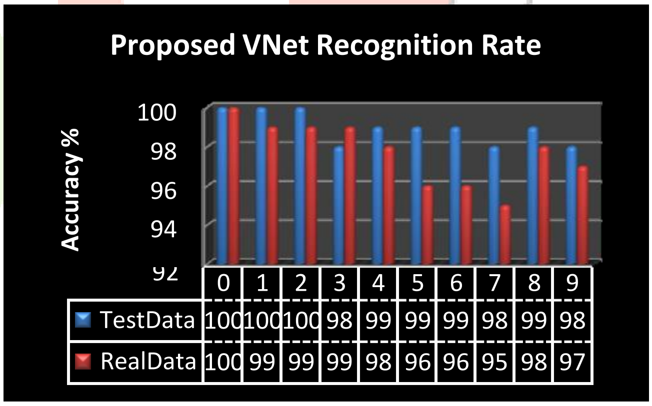

Fig.6 Recognition rate of proposed system for 0 to 9 digits

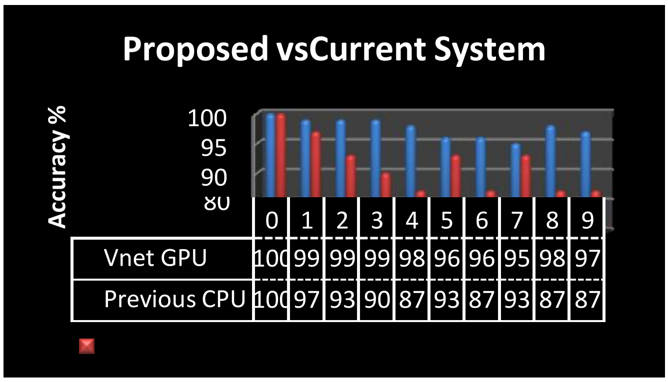

Fig.7 Comparison with existing system recognition rate for hand written digits

6. CONCLUSION AND FUTURE SCOPE

In this work we introduced quick proficient fake neural organization for written by hand digit acknowledgment on GPU to decrease preparing time with PTM (Equal Preparing Technique). We inferred back proliferation calculation on GPU based parallelization ought to be favored by and large with contrasted with computer chip based program. Yet, for back proliferation with little info information and few secret neurons computer chip based execution is better. Be that as it may, if the information dataset is bigger than GPU based parallelization is appropriate to lessen preparing time.

We hoping to investigate the capacities of GPU and multi cores in cloud conditions and offering them as administrations, where clients can question and choose them, contingent upon individual help level arrangements and the framework is giving superior via programmed 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] 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.