2017 IEEE 3rd International Conference on Sensing, Signal Processing and Security (ICSSS)

Parallelization Of Digit Recognition System Using Deep Convolutional Neural Network On CUDA

Srishti Singh
School of Electronics Engineering
VIT University
Vellore, India
srishtisingh3895@gmail.com

Amrit Paul
School of Electronics Engineering
VIT University
Vellore, India
paul_amrit@yahoo.co.in

Dr. Arun M
Associate Professor, School of Electronics Engineering
VIT University
Vellore, India
arunm@vit.ac.in

***Abstract*— A Compute Unified Device Architecture (CUDA) implementation of Deep Convolutional Neural Network (DCNN) for a digit recognition system is proposed to reduce the computation time of ANN and achieve high accuracy. A neural network with three layers of convolutions and two fully connected layers is developed by building input, hidden and output neurons to achieve an improved accuracy. The network is parallelized using a dedicated GPU on CUDA platform using Tensorflow library. A comparative analysis of accuracy and computation time is performed for sequential and parallel execution of the network on dual core (4 logical processors) CPU, octa core (16 logical processors) only CPU and octa core (16 logical processors) CPU with GPU systems. MNIST (Modified National Institute of Standards and Technology) and EMNIST (Extended MNIST) database are used for both training and testing. MNIST has 55000 training sets, 10000 testing sets and 5000 validation sets. EMNIST consists of 235000 training, 40000 testing and 5000 validation sets. The network designed requires high computation and hence parallelizing it shows significant improvement in execution time.**

***Keywords*— CUDA, cuDNN, Tensorflow, GPU, ANN, CNN, Back Propagation, Supervised Learning, Parallel Computing, Pattern Recognition, API**

I. INTRODUCTION

The application of machine learning is redefining today's technology, hence artificial neural networks are being widely employed for image, video, text, facial recognition etc., and in many other fields. A multi-layered convolutional neural network design is proposed in this paper to implement the digit recognition system with a high accuracy rate. The training of an Artificial Neural Network (ANN) is a time-consuming process [6] in machine learning systems and the computation time increases with the number of layers. Hence parallelizing the process using a dedicated graphics processing unit (GPU) will improve the time factor. CUDA (Compute Unified Device Architecture), a parallel computing platform and programming model invented by NVIDIA, enables dramatic increase in computing performance by harnessing the power of the GPU. This can be implemented by using Tensorflow library, which is an open source software library for machine learning developed by Google. Its flexible architecture facilitates the deployment of computation to one or more CPUs or GPUs. Thus, CUDA implementation on digit recognition system using Tensorflow and deep convolutional neural network can improve both accuracy as well as computation time.

Digit recognition can be achieved with high accuracy using deep convolutional neural networks. In which the image is fed to convolution filters and then its output to small neural networks. This combination of convolution filters and neural networks form the convolutional neural networks. This stage is repeated in the system to increase the accuracy of recognition. This multi-layered architecture of this network constitute the deep convolutional neural network (DCNN). Max-pooling block is used when the down-sampling of size is required. The fully connected neural networks are used at the end for the final prediction of the digit. The back propagation neural networks (BPN) are used as fully connected networks (FCNs). In the system, rectified linear units (ReLU) are employed to convert any negative value of pixel to zero. So that only zero and positive pixels exist. For decreasing the computation time, parallel computing is done on this DCNN on CUDA platform.

GPU hardware and user friendly APIs for parallel computing like CUDA provides affordable, programmable and high-performance computing environments for different applications including the parallelization of artificial neural networks executions. There are a lot of research presenting new approaches to handwritten digit recognition, neural networks as well as parallel computing.

II. RELATED WORK

A lot of researches focus on accelerating the computation of ANNs by using parallel programming or a dedicated hardware for the execution. This results in better computation performance.

Pendlebury et al. [1] presented a program of parallel simulation of neural networks on NVIDIA CUDA. The simulation consisted of 128 mine hunters in a minefield of 8192 mines. Intel Quad Core i5 3.3GHz 2xNvidia GeForce GTX 480 were used. 80% improvement in the performance was observed when execution was done with CUDA instead of CPU.

Shunlu Zhang et al. [5] implemented parallel Back-Propagation neural network (BPN) training using CUDA on multiple GPUs. Batch mode BPN training was proposed to exploit the maximum performance of GPUs. Multiple GPUs were used to achieve further acceleration. All the GPUs used

978-1-5090-4929-5

same training networks and weight parameters but different datasets were provided to different GPU to achieve that. The results showed that the GPU training was 12 times faster than the CPU training and then multiple GPUs were used for training, the results were 51.35 times faster than CPU implementation.

Augusto Casas [6] parallelized the back propagation network by running the algorithm on four processors simultaneously. This resulted in a reduction of 61% in training time compared to the sequential execution of same algorithm. Honghoon Jang et al. [12] implemented neural-network based text detection system on both multi-core CPU and GP. This resulted in 15times faster execution than only CPU and 4 times faster than only GPU, without OpenMP.

There have been many researches to increase the accuracy of the recognition of digits. Euclidean minimum distance criterion is used by Ravi Babu et al. [8] to find minimum distances and k-nearest neighbor classifier to classify the digits. Recognition rate of 96.94% was obtained on 5000 numeral images as testing set from MNIST database.

Ernst Kussul et al. [10] developed a neural classifier Limited Receptive Area for the image recognition achieving a recognition rate of 99.4% when tested on MNIST dataset of hand written digits. But this led to a high training time of 55hrs on Pentium III, 500MHz processor. Cheng-Lin Liu et al. [11], implemented the state of the art feature extraction and classification techniques for the digit recognition on three different database. An accuracy of 99.42% was achieved on the MNIST dataset, but with high computation time on Pentium 1.7GHz processor.

Gregory Cohen et al. [13] provided the EMNIST database deriving it from NIST library and then processing it. They were able to achieve an accuracy of 97.22% on EMNIST digits dataset. They also performed the test for all other datasets of EMNIST and achieved an accuracy of 97.5% on EMNIST MNIST dataset.

III. DATABASE

A. MNIST

MNIST (Modified National Institute of Standards and Technology) database of handwritten digits is used in the proposed system. It consists of 55000 training set samples, 10000 testing set samples and 5000 validation set samples. This contains the size-normalized images which fit into 20x20 pixel box, where the aspect ratio is kept constant. And then are centered into a 28x28 pixel box by computing the center of mass of the pixels. Original black and white images are converted to gray scale by anti-aliasing technique during the normalization process.

The MNIST (Modified NIST) database is a subset of the datasets available from the NIST. It was constructed with the combination Special Dataset 3 (SD3) and Special Dataset 1 (SD1) of NIST. SD3 contains cleaner digits than SD1 and hence the combination was used. 30000 samples from both the datasets were used to form the 600000 samples of training. And testing set has 5000 samples from both the sets. The first 5000 samples of the test set were taken from the original NIST training set and the last 5000 were taken from the NIST test set. The 5000 samples from MNIST training set are used for validation. Thus, 55000 samples are used for training, 10000 for testing and 5000 for validation. The data is stored in a simple file format designed for storing vectors and multidimensional matrices. It provides training dataset images, labels and testing data set images as well as labels. Fig.1 provides the sample images of handwritten digits and labels from MNIST database.

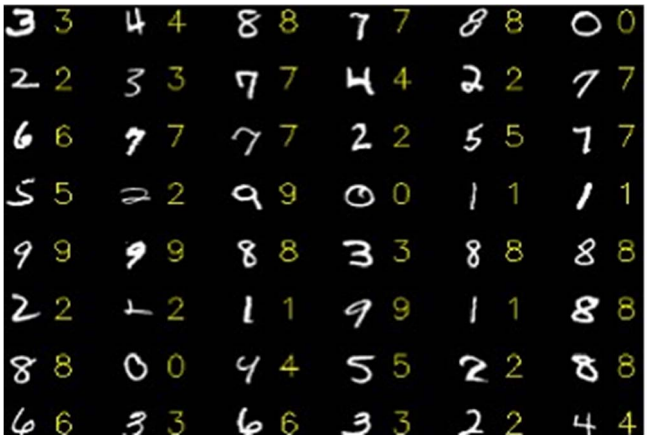

Figure 1. MNIST database sample images with labels

B. EMNIST

The EMNIST dataset provides both handwritten digits and characters, constructed from NIST Special Database 19 (SD19). The images are converted to a 28x28 pixel format similar to that of MNIST. It has a total of 280000 characters as EMNIST Digit dataset, with 10 unbalanced classes. It provides 235000 training dataset samples, 40000 testing samples and 5000 validation set samples.

It provides the data in two formats, both containing identical information. The first format provides it in a MATLAB format. And the second one provides it in the same binary format as the original MNIST dataset.

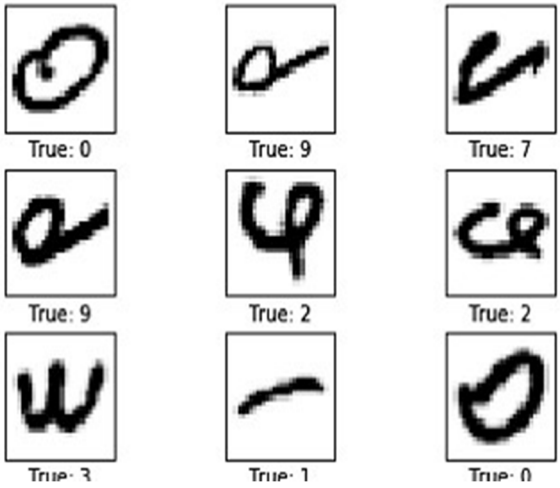

Figure 2. EMNIST database samples with true values

978-1-5090-4929-5©2017 IEEE

IV. ARCHITECTURE AND ALGORITHM

A system with good efficiency and less computation is required considering the huge computational time of ANNs training. A digit recognition system is developed using artificial neural network (ANN) consisting three convolutional neural networks (CNNs), two fully connected neural networks (FCNs), three max-pooling blocks (MPs), four rectified linear units (ReLUs) and one flatten layer. Hence constituting the deep convolutional neural network (DCNN), as deep learning uses the cascade of more than one neural networks.

DCNNs are used to increase the efficiency and accuracy of the system. But they increase the training time, since more computation is required as the layers and networks are increased. Tensorflow library is used to implement the DCNN on CPU as well as GPU, which is an open-source software library by Google. It is a powerful library for doing large-scale numerical computations. One of the tasks at which it excels is implementing and training deep neural networks. It also provides support for the CUDA platform.

In the system developed, the input image set is given from MNIST library, which are (28x28) pixels centered and size-normalized images of hand written digits. These are fed to the CNN-1, which consists of 16 (5x5) pixels convolution filters, producing a combination of tiles of images. Then this combination of pixel tiles is fed to simple neural network generating an array of output. The output of this network is fed to MP block and then to ReLU. The MP block is used to downsize the data. Here, (2x2) max pooling is done, that is, in the grid fed to MP block, the one having the maximum value out of the 4 pixels (2x2) is retained and the rest are discarded. The max-pooled array is given to rectified linear unit (ReLU), which converts any negative pixel value to zero, so that only zero and positive pixels exist. It is an activation function checking the weighted sum with zero and is defined in (1), where α is the activation function value and x is the input. Then the output of (14x14) pixels is generated having 16 image tiles.

$$\alpha = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \quad (1)$$

This image is fed to CNN-2, which has 36 (5x5) filters in the output channel, thus 576 (36x16) filters in total and again the output of this CNN is given to (2x2) MP block and a ReLU. After this, a (7x7) x 36 images are generated which are fed to CNN-3, which has 64 (5x5) filters in output channel, hence making it 2304 (36x64) filters in total. Again, (2x2) max pooling is performed for the output from CNN-3 and then is passed through a ReLU, generating 64 tiles of (4x4) pixels image output. The output pixel box is halved each time because of the (2x2) max-pooling.

These outputs are of the 4-D tensors form as the Tensorflow library is used, which needs to be converted to 2-D tensors for further use. 4-D tensors defines the 4 parameters which are input, filter, strides and padding. Hence, the output of the CNNs is sent to the flatten layer, which converts the 4 dimensional tensor to 2 dimensional tensor. Now the output has to be fed to FCNs. The CNN filters out the garbage and only useful data exists at the end. FCNs are simple fully connected neural networks where the actual prediction takes place. Back-propagation networks are used as FCN. The 64 x (4x4) images for is converted to a one dimensional array of size 1024. Hence, the FCN requires 1024 input neurons. In the final layers, softmax function and cross entropy calculation are used for getting better output results. Softmax is an activation function having the property that the resulting values sum up to 1, hence making it a better choice for normalizing the output as multiclass classifiers are used. Equation (2) gives the activation function value for each element (x_i) for a sequence $x = [x_1, x_2, x_3, \ldots, x_n]$.

$$\text{Softmax}(x_i) = x_i / \sum x_k \quad (2)$$

The cross entropy gives the measurement of the distance between the predicted output and actual output. Hence, less discrepancy will result in better performance. Even if the networks have same classification accuracy, the one with better performance can be calculated. The cross entropy is calculated as given in (3) for two different probability distributions, p and q here.

$$\text{Cross}(p,q) = -\sum p(x) \ln(q(x)) \quad (3)$$

Generalised form of cross entropy calculation is given in (4) with cross entropy as E_n, for n element vectors with target values $\tau = [t_1, t_2, \ldots, t_n]$ and output $y = [y_1, y_2, \ldots y_n]$.

$$E_n = -(1/n) \sum [\, t_k \ln(y_k) + (1 - y_k) \ln(1 - y_k) \,] \quad (4)$$

A combination of two fully connected neural networks is used in the system. The FCN-1 contains 1024 input neurons and 128 output neurons. There is a ReLU after FCN-1 to correct negative pixels. FCN-2 contains 128 input neurons and 10 output neurons, each output neuron corresponding to a digit. The input to FCN-1 is fed directly from the output of flatten layer and output of FCN-1 is sent to FCN-2. The final output corresponds to one of the values from 0-9.

The algorithm developed is executed on dual core (4 logical processors) CPU, having Intel core i3 processor 1.7GHz, 4GB RAM and Ubuntu 14.04. It was also executed on octa core (16 logical processors) CPU, having Intel Zeon processor 2.1GHz, 32GB RAM and Windows 10, 64-bit. The parallel execution was done on CUDA v8.0 platform using cuDNN v5.1 libraries with NVIDIA Quadro M4000 GPU hardware. All the executions were done using the Tensorflow libraries.

V. RESULTS

The purpose of this work was to measure and analyze the improvement in the computation time to implement the deep CNN for the digit recognition system and also achieving a high recognition rate.

The highest accuracy achieved for MNIST database is 99.49%. And the computation time for this was 6mins 33secs, which was achieved using the 3 CNNs on GPU with 30000 iterations. The highest accuracy achieved for EMNIST digits dataset is 99.62% with the computation time of 8mins 56secs, using 5 layered DCNN on GPU with 30000 iterations. Table I provides the values of computation time for both MNIST and EMNIST datasets, for 30000 iterations on all the three systems. Fig.3 shows the comparison of computation time in different systems.

978-1-5090-4929-5©2017 IEEE

TABLE I. COMPUTATION TIMES

Database	Computation time (hh:mm:ss)		
	CPU- dual core	*CPU- octa core*	*GPU*
MNIST	03:07:28	01:34:22	00:06:33
EMNIST	03:09:08	02:05:52	00:08:56

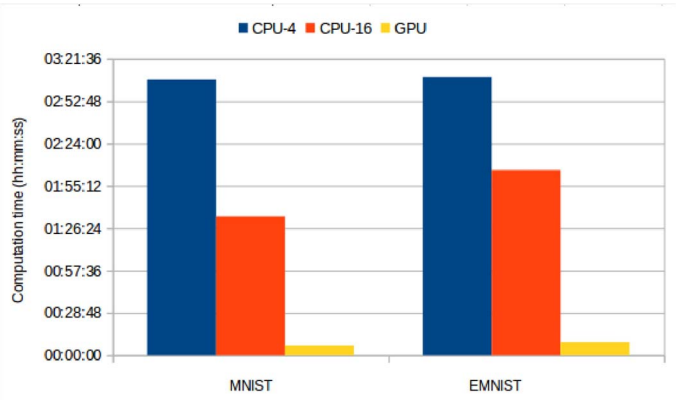

Figure 3. Comparison of computation time

Table II shows the maximum accuracies obtained on MNIST and EMNIST datasets for different number of iterations on GPU, keeping the number of CNNs 3 for MNIST dataset and 5 CNNs for EMNIST dataset.

TABLE II. ACCURACY AND ITERATIONS

Iterations	Accuracy (%)	
	MNIST	*EMNIST*
10000	99.26	99.42
20000	99.36	99.54
30000	99.49	99.62
40000	99.34	99.50
50000	99.31	99.54

We observe that an accuracy of 99.49% is achieved using 3 CNNs for MNIST Dataset and an accuracy of 99.62% is achieved using 5 CNNs for EMNIST Dataset. As the number of iterations increases, the computation time increases proportionally, but doesn't improve the accuracy. Therefore we restrict ourselves to 30000 iterations to achieve the optimum performance.

Fig.4 shows the variation of accuracy with number of iterations for both MNIST and EMNIST datasets.

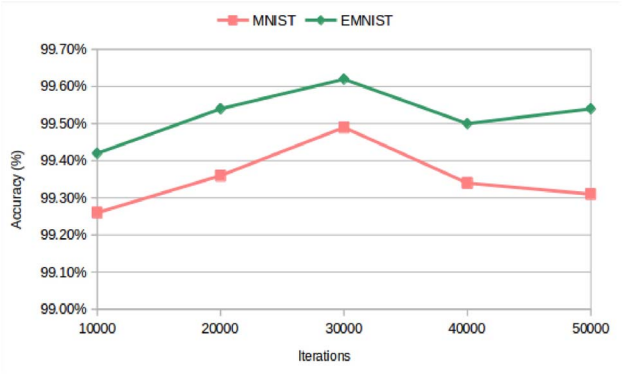

Figure 4. Variation of accuracy with iterations

Table III shows accuracies obtained on MNIST and EMNIST datasets as the number of CNNs are changed, keeping the number of iterations equal to 30000.

TABLE III. ACCURACY AND DEPTH

Depth	Accuracy (%)	
	MNIST	*EMNIST*
1	98.99	99.40
2	99.42	99.55
3	99.49	99.52
4	99.40	99.46
5	99.42	99.62

Fig.5 shows the variation of accuracy with depth of the system, where depth is the number of convolutional neural networks in the DCNN system.

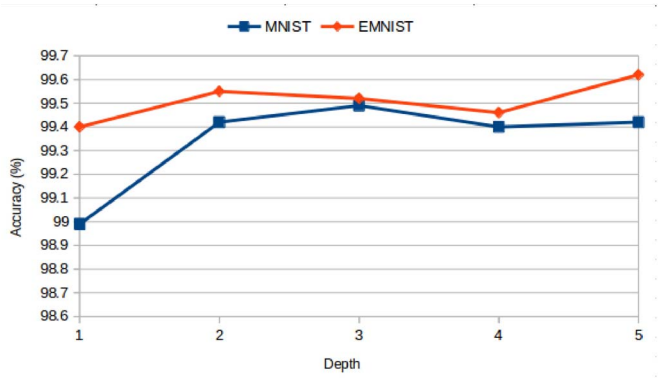

Figure 5. Variation of accuracy with depth

VI. CONCLUSIONS

The results of this paper show that the maximum accuracy of 99.49% for MNIST dataset was obtained at 30000 iterations and 3 CNNs deep learning system. And 99.62% for the EMNIST dataset with 30000 iterations and 5 layered DCNN. It is observed that accuracy increases with the number of iterations till some point (30000 here) and then it starts to fall back. As the number of CNN layers are increased the accuracy

978-1-5090-4929-5©2017 IEEE

rises to peak and then falls or after adding some more layer increases a little, for MNIST it increased till 3 layers and then started to decrease, but for EMNIST initially it increased from 1 to 2, then fell, but then again rose giving the maximum value of accuracy with 5 CNN layers.

It is also concluded that using the GPU hardware drastically decreases the computation time by 96.51% for MNIST and 95.27% for EMNIST than dual core CPU. That is, the execution on GPU is 28.62 times faster than dual core CPU for MNIST and 21.18 times faster for EMNIST computation.

The future work of the paper includes the preprocessing of the input data for DCNN to further increase the accuracy of recognition. Moreover, the computation performance of the system can be improved by using multiple GPUs for the execution.

REFERENCES

[1] Pendlebury, John, Huanhuan Xiong, and Ray Walshe. "Artificial neural network simulation on CUDA." *Proceedings of the 2012 IEEE/ACM 16th International Symposium on Distributed Simulation and Real Time Applications*. IEEE Computer Society, 2012.

[2] Devireddy, Srinivasa Kumar, and Settipalli Appa Rao. "Hand written character recognition using back propagation network." *Journal of Theoretical and Applied Information Technology* 5.3 (2009).

[3] LeCun, Y., et al. "Handwritten digit recognition with a back-propagation network, 1989." *Neural Information Processing Systems (NIPS)*.

[4] Sharma, Ankit, and Dipti R. Chaudhary. "Character recognition using neural network." *International Journal of Engineering Trends and Technology (IJETT)* 4.4 (2013): 662-667.

[5] Zhang, Shunlu, Pavan Gunupudi, and Qi-Jun Zhang. "Parallel back-propagation neural network training technique using CUDA on multiple GPUs." *Numerical Electromagnetic and Multiphysics Modeling and Optimization (NEMO), 2015 IEEE MTT-S International Conference on*. IEEE, 2015.

[6] Casas, C. Augusto. "Parallelization of artificial neural network training algorithms: A financial forecasting application." *Computational Intelligence for Financial Engineering & Economics (CIFEr), 2012 IEEE Conference on*. IEEE, 2012.

[7] Gupta, Roopali, and Neeraj Shukla. "Character Recognition using Back Propagation Neural Network." *International Journal of Digital Application & Contemporary research* (2012).

[8] Babu, U. Ravi, Y. Venkateswarlu, and Aneel Kumar Chintha. "Handwritten digit recognition using K-nearest neighbour classifier." *Computing and Communication Technologies (WCCCT), 2014 World Congress on*. IEEE, 2014.

[9] Hosseinzadeh, Hamidreza, and Farbod Razzazi. "Domain adaptive multiple kemel learning for handwritten digit recognition." *Electrical Engineering (ICEE), 2016 24th Iranian Conference on*. IEEE, 2016.

[10] Kussul, Ernst, and Tatiana Baidyk. "Improved method of handwritten digit recognition tested on MNIST database." *Image and Vision Computing* 22.12 (2004): 971-981.

[11] Liu, Cheng-Lin, et al. "Handwritten digit recognition using state-of-the-art techniques." *Frontiers in Handwriting Recognition, 2002. Proceedings. Eighth International Workshop on*. IEEE, 2002.

[12] Jang, Honghoon, Anjin Park, and Keechul Jung. "Neural network implementation using cuda and openmp." *Computing: Techniques and Applications, 2008. DICTA'08. Digital Image*. IEEE, 2008.

[13] Cohen, Gregory, et al. "EMNIST: an extension of MNIST to handwritten letters." *arXiv preprint arXiv:1702.05373* (2017).

978-1-5090-4929-5©2017 IEEE