LITERATURE REVIEW: PARALLEL PROCESSING OF HANDWRITTEN CHARACTER RECOGNITION

Shreya Patel
School of Computer Science
Carleton University
Ottawa, Canada K1S 5B6
shreyapatel4@cmail.carleton.ca

October 7, 2022

1 Introduction

Handwritten character recognition is the process of recognizing the unique handwriting of human beings and converting them into digital form. It usually involves converting written characters from sources like images, files, and documents into a machine-readable form. Convolutional Neural Network Architectures have proved to be quite efficient for Handwritten Character Recognition by learning the distinguishable traits from large datasets and recognizing the written text by understanding the shapes and characteristics of the letters. However, there is a drawback to the traditional CNN architecture. Since handwriting is a unique characteristic of each individual there is a wide spectrum of available data to process and thus the large computational requirement of neural networks regards for large computing time. The research proposes parallelizing the architecture to future improve the computational speed. There is a wide scope of exploring the parallel and distributed computing capacities of the dedicated hardware in order to achieve high accuracy of such recognition algorithms by training the neural nets for a huge amount of data.

1.1 Parallel Computing using GPU

Neural Networks are huge parallel structures made up of a large number of similar units that carry out the same calculation on various types of data. The majority of these units don't depend on one another for data, thus they can all compute simultaneously. On a typical CPU cluster, training such a network is challenging because of the large amount of data being transferred and processed between the units. If the entire network can run on a single GPU rather than a CPU cluster communication latency is minimized, bandwidth is increased, and size and power consumption are significantly reduced which decreases the computational efforts. The most fundamental distinction between GPU and CPU architectures is that the CPU cluster operates under the Multiple Instruction Multiple Data (MIMD) regime, where a number of independent processors execute various instructions on various data, whereas CUDA enabled GPU architectures operate under the Single Instruction Multiple Data (SIMD) regime, where a number of processors execute the same instruction on various data.

1.2 Compute Unified Device Architecture CUDA

The project suggests a multi-stream concurrent and parallel GPU-based approach for handwritten character recognition. It proposes constructing a programming environment convolutional neural network algorithm based on the CUDA architecture. GPU can efficiently speed up the CNN algorithm as compared to conventional microprocessors. This network can be parallelized by the means of a dedicated GPU using the CUDA platform. CUDA is a parallel computing platform and programming model that was developed by NVIDIA. It can be implemented using the TensorFlow Library. It is capable of achieving significant improvement in computing speed and efficiency by utilizing the GPU's capability.

1.3 Dataset

EMNIST[2] dataset can be used for the proposed system. The NIST[3] Special Database 19 was used to develop the EMNIST dataset, which offers both handwritten digits and characters. The images are transformed into a 28x28 pixel format similar to MNIST [1]. The EMNIST Digit dataset contains a total of 280000 characters in 10 unbalanced classes. It contains 235000 training dataset samples, 40000 testing samples and 5000 validation set samples. It offers the data in two different formats, having similar information. It is offered in MATLAB format in the first format. The second one is in a binary format similar to the primary MNIST dataset. The characters have been size-normalized and centred in the image which helps reduce the pre-processing to a great extent.

2 Literature Review

Several researches have been conducted over the years proving that parallel implementation of character recognition systems using the deep convolutional neural network can improve both accuracies as well as computation time to a great extent.

Augusto Casas[4] and S. Shunlu Zhang et al.[8] have proposed parallelization of Back Propagation neural networks that not only accelerates the computation of the Neural Network but also results in better computation performance. S.Shunlu Zhang et al. trained the Back Propagation neural network on multiple GPUs using CUDA and used batch mode training on multiple GPUs. They obtained an increase in speed of 12 times for a single GPU and 51.35 times for multiple GPUs than the CPU implementation. Augusto Casas simultaneously ran the algorithm on four CPUs. Compared to the sequential execution of the same algorithm, this led to a 61% reduction in training time.

The research[6] proposed reducing the run time of the Neural Network algorithms by the implementation of CUDA-enabled parallel algorithms on GPUs. Izotov et al. introduced a network of input, two convolutional and two fully connected layers and proposed converting the sequential algorithm into parallel by replacing loops with parallel SIMD-type instructions to perform fast vector and matrix multiplications operations using the CUBLAS functions. They obtained 8.6 times increased recognition rate for the CUDA-enabled software implemented on GPU than a single core CPU. However, the test error rate for GPU trained networks is 1.6% which is quite more as compared to the test error rate of 1.3% for the CPU trained network.

A similar observation in accuracy was noted by[7] when they increased the number of iterations over 30000 with GPU. They achieved the highest accuracy of 99.49% by implementing 3CNN on GPU with 30000 iterations. The computational time was around 6

mins. But when the iterations were increased above 30000, though the computation time decreased proportionally, the accuracy was affected drastically. They also noted that adding more layers to the CNN network gave a notable increase in the accuracy to a certain number of layers. They found maximum accuracy for 5 layered CNN. They were also able to achieve a 96.5% increase in the computation time by implementing on the parallel architecture as compared to a dual core CPU.

G. Du et al.[5] took into consideration the parallelization characteristics of matrix operations to use CUDA programming to implement the convolutional neural network algorithm on GPU. They showed that such implementation has the advantages of fast convergence, precise recognition rate and fast computational speed. They used the cublasSgemm function to perform matrix calculations in parallel. They took into consideration batch processing and performed a total of 2400 iterations on a batch of 50 data to achieve an accuracy of 97.59% for a run time of 75.12 seconds.

References

[1] The mnist database.

[2] The emnist dataset, Mar 2019.

[3] Nist special database 19, Apr 2019.

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

[5] Gaili Du, Liwei Jia, and Li Wei. A new algorithm of handwritten numeral recognition based on gpu multi-stream concurrent and parallel model. *2020 IEEE 2nd International Conference on Civil Aviation Safety and Information Technology (ICCASIT*, 2020.

[6] P. Yu. Izotov, N. L. Kazanskiy, D. L. Golovashkin, and S. V. Sukhanov. Cuda-enabled implementation of a neural network algorithm for handwritten digit recognition. *Optical Memory and Neural Networks*, 20(2):98–106, 2011.

[7] Srishti Singh, Amrit Paul, and M. Arun. Parallelization of digit recognition system using deep convolutional neural network on cuda. *2017 Third International Conference on Sensing, Signal Processing and Security (ICSSS)*, 2017.

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