Int. J. Advance Soft Compu. Appl, Vol. 6, No. 2, July 2014
ISSN 2074-8523; Copyright © SCRG Publication, 2014

Machine Learning Big Data Framework and Analytics for Big Data Problems

Shafaatunnur Hasan[1], Siti Mariyam Shamsuddin[2], Noel Lopes[3]

[1,2]*UTM Big Data Centre*,
Universiti Teknologi Malaysia, Skudai, Johor
e-mail: shafaatunnur@gmail.com, mariyam@utm.my
[3]*UDI*, Polytechnic Institute of Guarda, Portugal
CISUC, University of Coimbra, Portugal
e-mail: noel@ipg.pt

Abstract

Generally, big data computing deals with massive and high dimensional data such as DNA microrray data, financial data, medical imagery, satellite imagery and hyperspectral imagery. Therefore, big data computing needs advanced technologies or methods to solve the issues of computational time to extract valuable information without information loss. In this context, generally, Machine Learning (ML) algorithms have been considered to learn and find useful and valuable information from large value of data. However, ML algorithms such as Neural Networks are computationally expensive, and typically the central processing unit (CPU) is unable to cope with these requirements. Thus, we need high performance computer to execute faster solutions such Graphical Processing Unit (GPU). GPUs provide remarkable performance gains compared to CPUs. The GPU is relatively inexpensive with affordable price, availability and scalability. Since 2006, NVIDIA provides simplification of the GPU programming model with the Compute Unified Device Architecture (CUDA), which supports for accessible programming interfaces and industry-standard languages, such as C and C++. Since then, General Purpose Graphical Processing Unit (GPGPU) using ML algorithms are applied on various applications; including signal and image pattern classification in biomedical area. The importance of fast analysis of detecting cancer or non-cancer becomes the motivation of this study. Accordingly, we proposed machine learning framework and analytics of Self Organizing Map (SOM) and Multiple Back Propagation (MBP) for big biomedical data classification problems. Big data such as gene expression datasets are executed on high performance computer and Fermi architecture graphical hardware. Based on the experiment, MBP and SOM with GPU - Tesla generates faster computing times than high performance computer with feasible results in terms of speed performance.

Keywords**:** ***GPGPU, big data framework, machine learning, soft computing, SOM, MBP, biomedical classification problems.***

1 Introduction

The volume of data being produced is increasing at an exponential rate due to our unprecedented capacity to generate, capture and share vast amounts of data. In this context, Machine Learning (ML) algorithms can be used to extract information from these large volumes of data. However, these algorithms are computationally expensive. Their computational requirements are usually proportional to the amount of data being processed. Hence, ML algorithms often demand prohibitive computational resources when facing large volumes of data. As problems become increasingly challenging and demanding (in some cases intractable by traditional CPU architectures), often toolkits supporting ML software development fail to meet the expectations in terms of computational performance. Therefore, the scientific breakthroughs of the future will undoubtedly be powered by advanced computing capabilities that will allow researchers to manipulate and explore massive datasets [1]. Somehow, the pressure is to shift development toward high-throughput parallel architectures, crucial for real-world applications. In this context, highly-parallel and programmable devices such as GPU can be used for general-purpose computing applications [2]. GPUs can provide remarkable performance gains compared to CPUs. Moreover, they are relatively inexpensive with affordable price, availability and scalability. Over the last few years, the number of GPU implementations of ML algorithms has increased substantially [3]. However, most of the implementations are not openly shared. The lack of openly available implementations is a serious obstacle to algorithm replication and application to new tasks and therefore poses a barrier to the progress of the ML field [4]. By using CUDA architecture, an open source GPU machine learning library (GPUMLib) was developed by Lopes and Ribeiro [3]. The aim is to provide the building blocks for the development of efficient GPU ML software. GPUMLib offers several advantages such as useful in adoption of soft computing methods particularly on the neural network algorithms and fast detection of errors. Moreover, most of the previous studies are focused on using Artificial Neural Networks (ANNs) for pattern recognition [5][6][7]. Hence, we proposed Soft Computing algorithms for big data problems, particularly in biomedical area. The aim is to provide fast analysis in detecting the cancer from non-cancer based on the extraction of useful information in gene expression, protein profiling and genomic sequence data. This study is also significant to women who have a high risk of ovarian cancer due to family or personal history of cancer [8]. The remainder of this paper is organized as follows: Section 2 discusses the previous studies on the development of Machine Learning methods such as ANN algorithms on graphical hardware. Section 3 provides explanation on machine learning big data framework and GPUMLib implementation; particularly on SOM and MBP. Section 4 presents the experimental setup, followed by experimental results and discussion in Section 5. Finally, a conclusion of the study will be discussed in Section 6.

2 Related Work

Early studies, Soft Computing algorithm with graphical hardware implementation has been proposed in game console application, supervised and unsupervised Artificial Neural Network (ANN) algorithms [6]. In 1998, Bohn started to implement SOM on Computer Graphics Interface (CGI) Workstation for computer graphic applications [9]. Later, Zhongwen et al. started to apply SOM algorithm with multi-pass method on commodity GPU's (ATI 9550 and Nvidia 5700) and INTEL P4 2.4G for CPU computing [10]. Campbell et al. proposed parameter-less SOM which eliminates the parameter of learning rate and neighborhood size [11]. Furthermore, SOM is also evaluated on GPU cluster to compute the scalability [12]. On the other hand, parallel implementation of SOM to observe the suitability for high dimensional problem has been implemented by [13][14]. In pattern classification, Kyoung-Su Oh and Keechul Jung applied Multilayer Perceptrom (MLP) for text detection [5]. Prabhu proposed unsupervised SOM for pattern classifier [15]. Meanwhile, Gadjos et al. applied unsupervised SOM for outage database [16]. Subsequently, combination of supervised and unsupervised SOM for image segmentation was introduced by Faro et al. [7]. Moreover, Takatsuka et al. applied the Geodesic SOM on standard machine learning dataset [17]. Their experimental results suggested that the GPU speed performance is not significant for small datasets such as iris, but is considerable on larger datasets (ionosphere and torus). In medical area, preliminary studies focused mainly on detecting the cancer nodule and non-nodule based on medical imagery [18]. In addition, Lopes and Ribeiro proposed parallel BP and MBP for Ventricular Arrhythmias (VAs) in biomedical applications [19]. The aim of their method was to equip fast detection of diseases which highly potential to sudden death.

Based on the previous study, there is still lacking of SOM-GPU implementation for high dimensional pattern analysis particularly on biomedical area. This is due to most of the studies, proposed feature selection process to cater the nature of dataset problems. Furthermore, high dimensional features and imbalance dataset have a great influence to the classification accuracy [20]. SOM is an algorithm for exploratory data analysis which provides mapping from high dimensional features to low dimensional features [21]. However, the distance calculation and searching for the Best Matching Unit (BMU) generally increases greatly the computational cost. Hence, we proposed a framework of machine learning big data analytics and the parallel implementation of SOM and MBP to speed up the computation time. The SOM and MBP with GPUMLib implementation will be discussed in the next section.

3 Machine Learning Big Data Framework and GPU Library (GPUMLib)

In this study, we propose machine learning big data framework as illustrated in Fig.1. This framework envisages the broad picture of machine learning in dealing with big data problems. The framework starts with the presentation of multi-structure input varieties from different sources, follow by the pipeline pre-processing phase prior to machine learning knowledge discovery. However, in this paper, we implement the parallelism on machine learning approaches of big data predictive knowledge discovery based on Neural Network (NN) algorithms; Multiple Backpropagation (MBP) and Self Organizing Map (SOM) using GPUMLib.

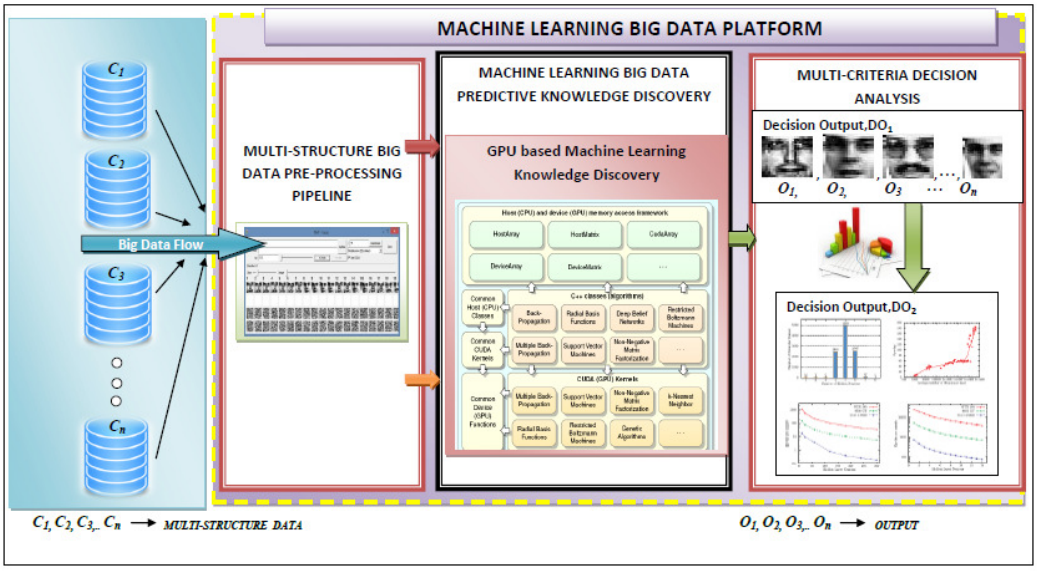

Fig. 1: Machine Learning Big Data Framework

MBP is an open-source algorithm built-in in GPUMLib [22]. Meanwhile, SOM algorithm is proposed in this study for the parallel implementation on the distance computation and BMU searching process. In the meantime, the parallelism on SOM algorithm uses the GPUMLib memory access and reduction frameworks. The GPUMLib memory access framework contains *HostArray, HostMatrix, DeviceArray, DeviceMatrix and CudaArray* classes. The framework manages to allocate the memory on the host and device, transfer data between host to device and vice versa. In the reduction framework, the *MinIndex* Kernel is designed to compute the minimum of an array and its corresponding index within the array. Both algorithms use batch training for parallel implementation and will be explained in section 3.1 and 3.2 respectively.

3.1 Parallel Multiple backpropagation (MBP)

MBP networks are designed based on multiple feed-forward architecture. They differ from standard BP networks, as they integrate two networks designated by main and space networks. The main network contains selective activation neurons which determine their importance for the actual *stimuli* from the space network. Therefore, the selective activation neurons choose and respond to specific group of patterns based on the input presented to the main network. Consequently, fine-tuning the network response according to the actual space localization features.The main network only calculates its outputs after space network outputs are evaluated. The implementation relies in five kernels: *FireLayer, FireOutputLayer, CalculateLocalGradient, CorrectWeights* and *CorrectOutputWeights* which execute in each epoch [22]. Initially, *FireLayer* and *FireOutputLayer* kernels are launched by the host in order to determine the space and main network output. Consequently, the main network weights are adjusted using the parallelism of *CalculateLocalGradient, CorrectWeights* and *CorrectOutputWeights* kernels. Finally, the space network weights are adjusted with *CorrectOutputWeights* kernel. In addition, an Autonomous Training System (ATS) is implemented to improve MBP result. The ATS train several MBPs to select an appropriate MBP network topology. As new MBP networks are trained, its performance is compared with the best MBP found so far. These results are then used to determine the number of hidden neurons of a new MBP and adjusted accordingly until the termination criterion is satisfied [23].

3.2 Parallel Self Organizing Map (SOM)

The SOM implementation, developed in this study, using the GPUMLib is executed on GPU (host and device) and CPU (host only). For better representation, the implementation is depicted in Fig 2. Basically, the input data and the weights are initialized randomly on the host side. Meanwhile, the Best Matching Unit (BMU) searching is implemented on the device side. In this process, the memory is allocated for the both side (host and device) and also transfer from host to device (vice versa). For instance, the weights and input data function variables are defined in a *HostMatrix* (host side) and in a *DeviceMatrix* (device side). Next, the ComputeDistanceskernel<<<⋯>>>, depicted in Fig. 3 is launched. This function is designed purposely to calculate the sum squared distance between the input data and weights, i.e. the *Euclidean* distance. Subsequently, the reduction framework, *MinIndex* Kernel is launched (See Fig 4). The reduction process synchronizes the threads, in order to find the minimum value of BMU (*x, y*). Consequently, the result of each block is written to global memory. The minimum values are copied back to the host for updating the weights. Hence, the looping process continues until the termination criterion is satisfied and finally displays the result. On the other hand, all the processes from read the input data to display output are fully executed on the host (CPU)

implementation. The distance and BMU are compute on BestMatchingUnit() function; without transfer to device (see Fig 2).

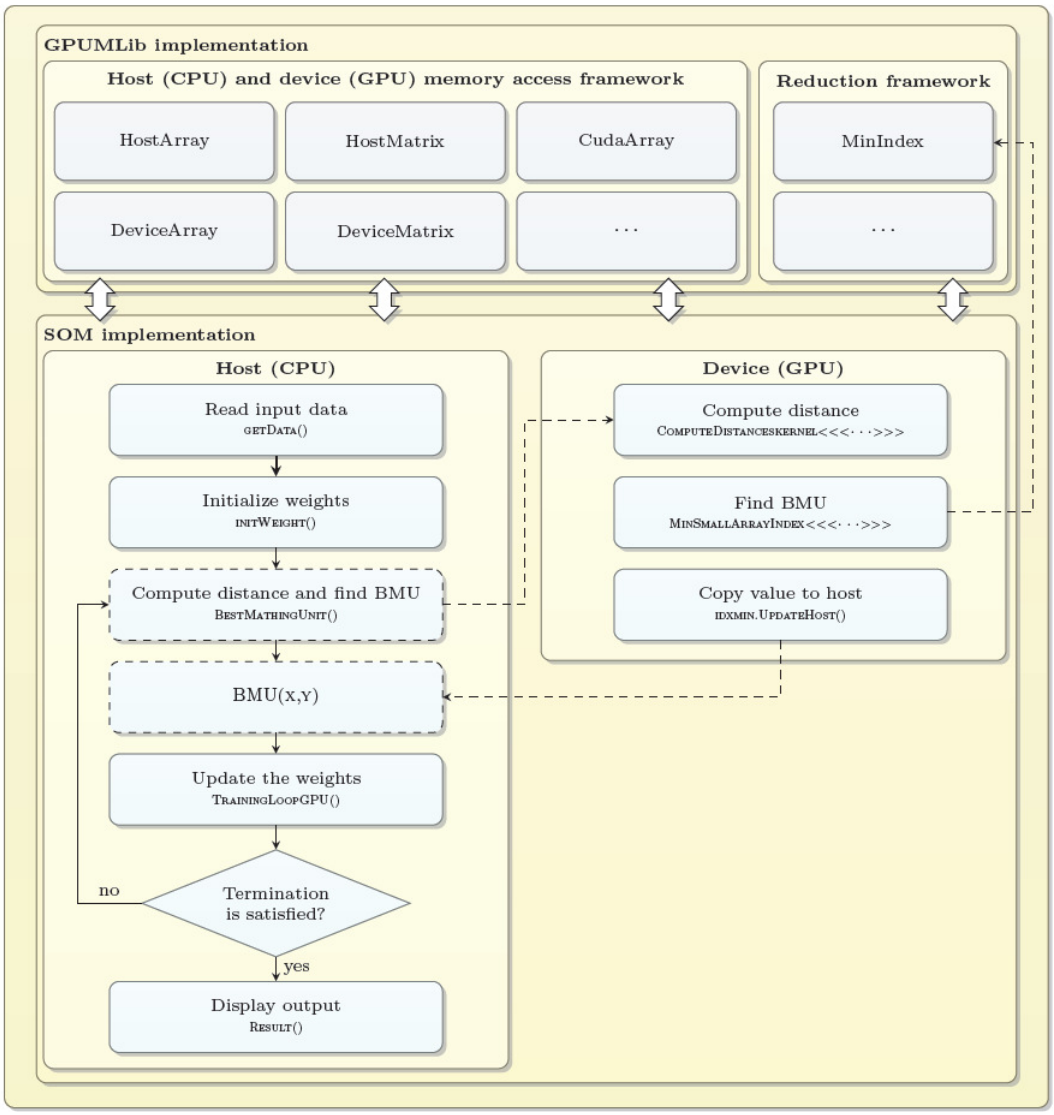

Fig. 2: SOM with GPUMLib Implementation on training the **Host (CPU)** and **Device (GPU)**

```
__global__ void ComputeDistancesKernel(float * inputData, float *
weights, int vector, int numberFeatures, float * distances) {
   extern __shared__ float sdist [];

   int i = blockIdx.x;
   int j = blockIdx.y;

   int w = i * gridDim.y + j; // weights have two dimensions

   float distance = 0.0;

   for (int feature = threadIdx.x; feature < numberFeatures;
   feature += blockDim.x) {
      float fdist = inputData[vector * numberFeatures + feature] -
                  weights[w * numberFeatures + feature];
      distance += fdist * fdist;
   }
   sdist[threadIdx.x] = distance;

   // reduction
   __syncthreads();

   for (int dist = blockDim.x; dist >= 2;) {
      dist /= 2;
      if (threadIdx.x < dist) {
        sdist[threadIdx.x] += sdist[threadIdx.x + dist];
      }
      __syncthreads ();
   }

   if (threadIdx.x == 0) {
      distances[w] = sqrt(sdist[0]);
   }
}
```

Fig. 3: Launching a kernel to compute distances

```
void KernelMinIndexes(cudaStream_t stream, int blocks, int
blockSize, cudafloat * inputs, cudafloat * output, int *
minIndexes, int numInputs, int * indexes) {
   MinSmallArrayIndex< blockSize ><<< blocks, blockSize, blockSize
   * (sizeof(cudafloat) + sizeof(int)), stream>>>(inputs, output,
   minIndexes, numInputs, indexes);
}
```

Fig 4: Launching a kernel to search the minimum value[1]

4 Experimental Setup

In this study, high dimensional biomedical dataset including gene expression data, protein profiling data and genomic sequence data that are related to

[1] http://gpumlib.sourceforge.net

classification is shown in Table 1. The Leukemia training dataset consists of 38 bone marrow samples which categorize as 27 Acute Myeloid Leukemia (ALL) and 11 Acute Lymphoblastic Leukemia (AML), over 7129 probes from 6817 human genes. Also 34 samples testing data are provided, with 20 ALL and 14 AML [24]. The prostate cancer training set contains 52 prostate tumor samples and 50 non-tumors which label as normal with 12600 genes. While, testing set consist of 25 tumor and 9 normal samples [25]. The proteomic patterns for ovarian cancer were generated by mass spectroscopy, which consists of 91 normal and 162 ovarian cancers. The raw spectral data of each sample contains 15154 identities and 253 samples [8]. All datasets are normalized within the range of 0 to 1.

Table 1: Biomedical Dataset[2]

No	Dataset	No. of Samples	No. of Features	Class Name
1	Leukimia	72	7129	ALL AML
2	Prostate Cancer	136	12600	Tumor Normal
3	Ovarian Cancer	253	15154	Tumor Normal

The SOM and MBP algorithms are executed on NVIDIA Tesla C2075 graphic hardware and Intel Xeon high performance computer. Both algorithms are tested on high dimensional biomedical datasets (Leukemia, Prostate Cancer and Ovarian Cancer). The SOM algorithm is setup for 1000 iterations in three different size of mapping. While the MBP algorithm executes for 10,000 iterations using the Autonomous Training System (ATS). Initially, MBP generates 100 networks with one and two hidden layers. The biomedical datasets such as Prostate cancer, ovarian cancer and leukemia dataset are indicated as large, medium and small feature dimensions. Meanwhile, the SOM mapping size (5x10, 10x10 and 10x15) are labeled as small, medium and large, respectively.

5 Experimental Result and Analysis

In this study, the aim of the analysis is to observe the capability of MBP and SOM algorithm using graphical hardware (GPU) on high performance computer (CPU). In this experiment, the size of SOM mapping dimension is categorized as mapping 1=5x10, mapping 2=10x10 and mapping 3= 10x15 (see Table 2). While, the number of hidden nodes are set to 100 for the first and a total 15 nodes for the

[2] http://datam.i2r.a-star.edu.sg/datasets/krbd/

second MBP network (see Table 3). Hence, large proportion size of mapping dimension, number of hidden nodes, iterations and feature dimensions of the dataset generates slow computation times for both algorithms. Since the computational time depends on certain parameters, we evaluate both algorithms with similar datasets, number of nodes and number of iterations. Moreover, the speed performance is calculated based on CPU time and GPU time, respectively.

As a result, the SOM speed on GPU generates approximately three times speed up compare to CPU for all datasets as depicted in Fig. 5. Meanwhile, MBP produce significant performance with 27 times speed up for 10,000 iterations on Leukemia dataset (CPU time= 838s and GPU time=30.654s). Otherwise, SOM (Size of Mapping = 10x10) generates 533.726s on CPU; 12 times speed up than MBP network (100 nodes) with 6612s for 1000 iterations (see Table 4).

Table 2: SOM (host and device) Speed Performance

Dataset	Performance Evaluation	SOM Result		
Leukemia	**No. of Iteration**	1000		
	Size of Mapping	**Mapping 1**	**Mapping 2**	**Mapping 3**
		5x10	**10x10**	**10x15**
	CPU time	356.436 s	533.726 s	974.418 s
	GPU time	115.441 s	207.574 s	301.79 s
	Speed	3.088 x	2.571 x	**3.229 x**
Prostate Cancer	**No. of Iteration**	1000		
	Size of Mapping	**Mapping 1**	**Mapping 2**	**Mapping 3**
		5x10	**10x10**	**10x15**
	CPU time	1621.65 s	2618.41 s	4081.941 s
	GPU time	660.474 s	1118.06 s	1565.038 s
	Speed	2.455 x	2.341 x	**2.608 x**
Ovarian Cancer	**No. of Iteration**	1000		
	Size of Mapping	**Mapping 1**	**Mapping 2**	**Mapping 3**
		5x10	**10x10**	**10x15**
	CPU time	3455.925 s	6354.214 s	9116.06 s
	GPU time	1086.895 s	2061.42 s	3166.077 s
	Speed	**3.180 x**	3.082 x	2.879 x

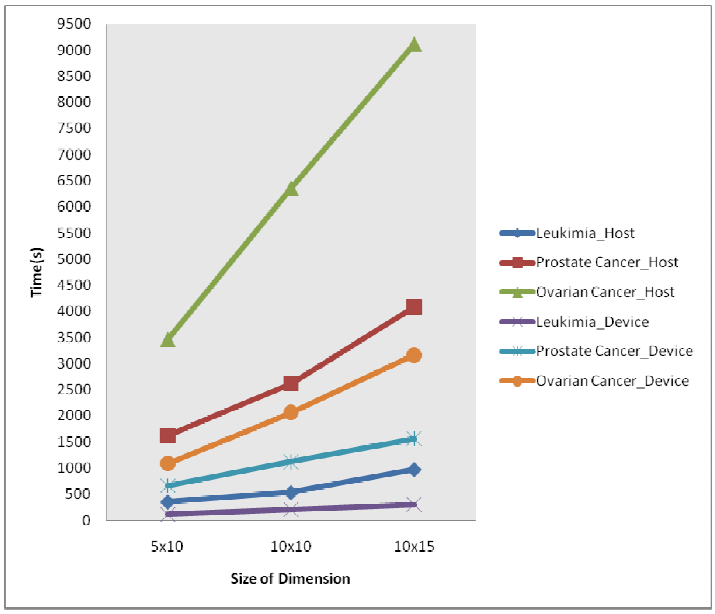

Fig. 5: SOM speed Analysis

Table 3: MBP-GPU (device) Speed Performance

Dataset	Performance Evaluation	MBP Result			
Leukemia	**MBP Network**	**7129-100-1**		**7129-5-10-1**	
	Min/Max Iteration	**Min**	**Max**	**Min**	**Max**
	No. of Iteration	146	163	230	10000
	GPU time	8.892s	9.928s	**0.702s**	30.654s
Prostate Cancer	**MBP Network**	**12600-100-1**		**12600-5-10-1**	
	Min/Max Iteration	**Min**	**Max**	**Min**	**Max**
	No. of Iteration	181	225	2996	10000
	GPU time	36.042s	44.805s	**23.678s**	79.119s
Ovarian Cancer	**MBP Network**	**15154-100-1**		**15154-4-10-1**	
	Min/Max Iteration	Min	Max	Min	Max
	No. of Iteration	152	191	155	10000
	GPU time	77.173s	96.955s	**2.075s**	134.134s

Table 4: MBP-CPU (host) Speed Performance in Leukemia dataset

Performance Evaluation	**MBP Result**			
Iteration	1000	10000	1000	10000
MBP Network	7129-100-1		7129-5-10-1	
CPU time	6612s	48668s	109s	838s

6 Conclusion

In this study, we found that the results are proportionate to the mapping size of the SOM architecture and feature dimensions of the datasets. In other words, the larger the mapping size and feature dimensions, the slower the computation time for both CPU and GPU. This is due to ANNs (SOM and MBP) parameters that depend on size of mapping (number of nodes), dataset feature dimensions, number of input samples, and termination criterion (number of iterations or convergence rate). Our findings are conformed to the findings conducted by [12][14], i.e., larger mapping size will increase the memory transfer; thus slower the computational time [14]. The current GPU parallel implement of the SOM algorithm performs three times (3x) faster than the CPU, while the MBP is 27 times faster than the CPU. However, the SOM's speed could be improved with the parallelism on updating the weights. It is important for larger (big data) datasets that do not fit on the GPU memory, consists of devising methods to choose a representative subset of the data. Alternatively, we can also create several maps for different data that could afterwards be merged together latter in a bigger map. Furthermore, the aim of SOM-GPUMLib implementation will be openly shared in the future.

ACKNOWLEDGEMENTS
This work is supported by Universiti Teknologi Malaysia under Flagship Project: QJ130000.2428.02G38. The authors would like to thanks Research Management Centre (RMC), Universiti Teknologi Malaysia (UTM) for the support in R & D, and *Soft Computing Research Group* (SCRG) for the inspiration in making this study a success. The authors would also like to thank the anonymous reviewers who have contributed enormously to this work.

References

[1] Hey, T., Tansley, S., Tolle, K., editors (2009). The Fourth Paradigm: Data-Intensive Scientific Discovery. Microsoft Research.

[2] Owens, J. D., Houston, M., Luebke, D., Green, S., Stone, J. E., Phillips, J. C. (2008). GPU computing. Proceedings of the IEEE, 96(5):879–899.

[3] Lopes, N., Ribeiro, B., Quintas, R. (2010): GPUMLib: A new Library to combine Machine Learning algorithms with Graphics Processing Units. HIS 2010: 229-232

[4] Sonnenburg, S., Braun, M. L., Ong, C. S., Bengio, S., Bottou,L., Holmes, G., LeCun, Y., M¨ uller, K.-R., Pereira, F.,Rasmussen, C. E., R¨ atsch, G., Sch¨ olkopf, B., Smola, A.,Vincent, P., Weston, J., and Williamson, R. C. (2007).The need for open source software in machine learning.Journal of Machine Learning Research, 8:2443–2466.

[5] Kyoung S. H. Keechul J., (2004). GPU implementation of neural networks, Pattern Recognition, vol. 37, (6), pp. 1311-1314.

[6] Meuth, R., Wunsch, D. C. (2007). A Survey of Neural Computation on Graphics Processing Hardware. IEEE 22nd International Symposium on Intelligent Control, 2007. ISIC 2007.

[7] Faro, A., Giordano, D., Palazzo, S. (2012). Integrating unsupervised and supervised clustering methods on a GPU platform for fast image segmentation. 3rd International Conference on Image Processing Theory, Tools and Applications (IPTA), 2012

[8] Petricoin III, Emanuel F., Ali M. A., Ben A. H., Peter J. L., Vincent A. F., Seth M. S., Gordon B. M. et al. (2002): Use of proteomic patterns in serum to identify ovarian cancer. The lancet 359, no. 9306 572-577

[9] Bohn, C.A. (1998). Kohonen Feature Mapping Through Graphics Hardware. In Proceedings of 3rd Int. Conference on Computational Intelligence and Neurosciences.

[10] Zhongwen L., Hongzhi L., Zhengping Y., Xincai W., (2005) Self-Organizing Maps computing on Graphic Process Unit, in 1 3th European Symposium on Artificial Neural Networks, Belgium, pp. 557-562.

[11] Campbell, A., Berglund, E., Streit, A. (2005). Graphics Hardware Implementation of the Parameter-Less Self-organising Map. In M. Gallagher, J. Hogan & F. Maire (Eds.), Intelligent Data Engineering and Automated Learning - IDEAL 2005 (Vol. 3578, pp. 343-350): Springer Berlin Heidelberg.

[12] McConnell, S., Sturgeon, R., Henry, G., Mayne, A., Hurley, R. (2012). Scalability of Self-organizing Maps on a GPU cluster using OpenCL and CUDA. Paper presented at the Journal of Physics: Conference Series.

[13] Platos, J., Gajdos, P. (2010). Large data real-time classification with Non-negative Matrix Factorization and Self-Organizing Maps on GPU. International Conference on.Computer Information Systems and Industrial Management Applications (CISIM)

[14] Gajdoš, P., Platoš, J. (2013) GPU Based Parallelism for Self-Organizing Map, in Proceedings of the Third International Conference on Intelligent Human Computer Interaction (IHCI 2011), Prague, Czech Republic, August, 2011. vol. 179, M. Kudělka, et al., Eds., ed: Springer Berlin Heidelberg, pp. 231-242.

[15] Prabhu, R. D. (2008). SOMGPU: An unsupervised pattern classifier on Graphical Processing Unit, in Evolutionary Computation,CEC 2008. IEEE World Congress on Computational Intelligence, pp. 1011-1018.

[16] Gajdoš, P., Krátký, M., Bednár, D., Baca, R., Gono, R., Walder, J. (2011). Efficient Computation of SOM for Outage Database. ELNET 2011, 51.

[17] Takatsuka, M., Bui, M. (2010). Parallel Batch Training of the Self-Organizing Map Using OpenCL. In K. Wong, B. S. Mendis & A. Bouzerdoum (Eds.), *Neural Information Processing. Models and Applications* (Vol. 6444, pp. 470-476): Springer Berlin Heidelberg.

[18] Eklund, A., Dufort, P., Forsberg, D., LaConte, S. M. (2013). Medical Image Processing on the GPU - Past, Present and Future. Medical Image Analysis. Elsevier.

[19] Lopes, N., Ribeiro, B. (2009). Fast pattern classification of ventricular arrhythmias using graphics processing units. In Proceedings of the 14th Iberoamerican Conference on Pattern Recognition (CIARP 2009), LNCS 5856, pages 603–610. Springer.

[20] Tanwani, A., Farooq, M. (2009). The Role of Biomedical Dataset in Classification. In C. Combi, Y. Shahar & A. Abu-Hanna (Eds.), Artificial Intelligence in Medicine,Vol. 5651, pp. 370-374, Springer Berlin Heidelberg.

[21] Kohonen, T. (2001): Self-Organizing Maps. Springer Series in Information Sciences.Vol. 30. (3rd ed) Extended Edition. Springer-Berlin.

[22] Lopes, N., Ribeiro, B. (2009). GPU implementation of the multiple back-propagation algorithm. In Intelligent Data Engineering and Automated Learning-IDEAL 2009 (pp. 449-456). Springer Berlin Heidelberg.

[23] Lopes, N., Ribeiro, B. (2010). A strategy for dealing with missing values by using selective activation neurons in a multi-topology framework. The 2010 International Joint Conference on Neural Networks (IJCNN), pp. 1-5. IEEE.

[24] Golub, T. R., Donna K. S, Pablo T., Christine H., Michelle G., Jill P. M., Hilary C. et al. (1999). Molecular classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. Science 286, no. 5439: 531-537.

[25] Singh, D., Phillip G. F., Kenneth R., Donald G. J., Judith M., Christine L., Pablo T. et al. (2002). Gene Expression Correlates of Clinical Prostate Cancer Behavior. Cancer Cell 1, no. 2: 203-209.