Use of High-performance Graphics Processing Units for Power System Demand Forecasting

Ting He*, Ke Meng, ZhaoYang Dong**, Yong-Taek Oh† and Yan Xu***

Abstract – Load forecasting has always been essential to the operation and planning of power systems in deregulated electricity markets. Various methods have been proposed for load forecasting, and the neural network is one of the most widely accepted and used techniques. However, to obtain more accurate results, more information is needed as input variables, resulting in huge computational costs in the learning process. In this paper, to reduce training time in multi-layer perceptron-based short-term load forecasting, a graphics processing unit (GPU)-based computing method is introduced. The proposed approach is tested using the Korea electricity market historical demand data set. Results show that GPU-based computing greatly reduces computational costs.

Keywords: Artificial Neural Network, Graphics Processing Unit, Multi-layer Perceptron, Short-term Load Forecasting

1. Introduction

Load forecasting is one of the important requirements in the operation and planning of modern power systems. It is the basis of many significant tasks necessary to ensure power system stability and reliability [1]. For transmission network service providers (TNSPs), load forecasting is of great importance. Long-term increments in the demand for electricity trigger transmission line expansion planning, and short-term variations in demand determine daily operational strategies. For generators, an accurate future demand prediction is a key factor in setting up market strategies to secure the best bidding and bilateral contract positions in the competitive electricity market. For distribution network service providers, a reliable demand forecast is one of the most important factors in forming planning options, which are often made in conjunction with related TNSPs.

In terms of prediction period, load forecasting can be divided into long-term, mid-term, and short-term forecasting. Generally speaking, long-term forecasting is longer than one year, mid-term forecasting is from one week to one year, and short-term forecasting is from one hour to one week or a little longer. This paper deals with short-term load forecasting analysis. Given the importance of demand forecasts, various methods and tools have been developed and reported. These include the time series model [2], grey theory, regression analysis, neural network model [3]-[6], econometric demand forecasting, and composite model [7], [8]. Among these methods, the neural network model is well known for its powerful non-linear mapping and generalization abilities [9]-[12]. It has been widely implemented in electricity market demand forecasting, and it has been presented as an essential element of subsequent method developments.

To produce better demand forecasting results, aside from historical load data, other variables, such as weather or seasonal variations and holiday activities, should be considered as inputs in the neural network model. Large amounts of input variables not only complicate the neural network structure, but also increase the corresponding learning time. However, in scenarios where fast or even real-time load forecasting is required, the learning time of the neural network model is critical [13]. To overcome this time-consuming characteristic, most previous works on neural network optimization focused on the algorithmic level. In [14], the parallel dynamic learning concept was introduced as a different training methodology to achieve speedups. The primary advantage of this method is its transparency to architectural constraints; however, poor implementations may diminish speedups. The parallelism of neural networks using clusters and message-passing interface(MPI) also suffers from communication overheads and contention delays [15]. With the introduction of programmability, graphics processing units (GPU) have gained enough flexibility for use in non-graphic applications [16]. In this paper, the GPU-based parallel computing frame is introduced to accelerate hourly-ahead load forecasting with multi-layer perceptron (MLP).

The paper is organized as follows. After the introduction, MLP and GPU computing techniques are reviewed for completeness, followed by the specific steps of the proposed learning algorithm. The methodology is then tested with a demand series from the Korean electricity market. Conclusions and further developments are discussed in the last section.

† Corresponding Author: Korea University of Technology and Education, Korea
* School of ITEE, The University of Queensland, Australia
** Department of Electrical Engineering, The Hong Kong Polytechnic University, Hong Kong
*** Dept. of Industrial Management Engineering, Korea University, Korea. (rmyung@korea.ac.kr)
Received: January 18, 2010; Accepted: May 7, 2010

2. Multi-Layer Perceptron

As a computing system, an artificial neural network (ANN) is made up of a number of simple and highly interconnected processing units, which process information through dynamic state responses to external inputs. The basic unit of ANN is the artificial neuron, which receives information through a number of input nodes, processes them internally, and then produces associated responses through output nodes.

The neurons are organized in a way that defines the network architecture. The one we are most concerned with is MLP, in which the neurons are organized in layers. The architecture of typical MLPs is feed-forward; that is, the outputs of one layer are used as inputs to the following layer. The layers between the input layer and the output layer are called hidden layers.

2.1 Network Structure

The architecture of a multi-input and multi-output (MIMO) MLP with one hidden layer is presented in Fig. 1.

The above figure shows the configuration of a typical MIMO MLP. In MLP, each neuron is connected to other neurons in the next layer. The MLP output is described as

$$y_p = \varphi_O \left\{ \sum_{j=0}^{N} w_{jp}^{O} \left[\varphi_H \left(\sum_{i=0}^{M} w_{ij}^{H} x_i \right) \right] \right\} \tag{1}$$

where

w_{ij}^{H} is the input-hidden layer connecting weights;

w_{jp}^{O} is the hidden-output layer connecting weights;

φ_H is the activation function of the hidden layer; and

φ_O is the activation function of the output layer.

The objective of the learning method is to minimize MLP output error. To evaluate MLP performance, the root mean squared error (RMSE) is introduced.

$$RMSE = \sqrt{\frac{1}{N_h} \sum_{p=1}^{N_h} \left(y_p - \overline{y}_p \right)^2} \tag{2}$$

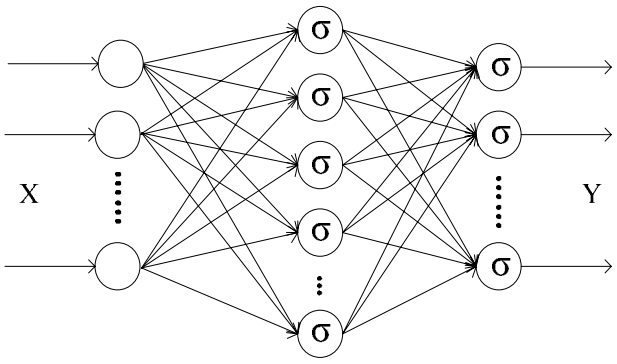


Fig. 1. Structure of a Typical MIMO MLP.

where

y_p is the true value; and

$\overline{y}_p$ is the predicted value.

2.2 Learning Methods

Demand forecasting with MLP involves training and testing. It is assumed that a training set is available; in this case, a historical data set, which contains inputs and corresponding desired outputs, is available. In the learning process, an MLP constructs an input-output mapping, adjusting the weights and biases at each iteration based on the minimization of error measurements between produced and desired outputs. Thus, learning entails an optimization process. The minimization process is repeated until an acceptable criterion is reached.

One of the most common learning methods is the Levenberg-Marquardt (LM) approach [17], a blend of gradient descent and Gauss-Newton iteration. The learning procedures can be summarized as follows:

$$\begin{cases} \nabla E(w) = J(w)^T e(w) \\ \nabla^2 E(w) \approx J(w)^T J(w) = H \end{cases} \tag{3}$$

To minimize the cost function,

$$Min. \nabla E(w) \to 0 \tag{4}$$

We then expand the gradient of E using the Taylor series around the current state:

$$w(t+1) = w(t) + \nabla^2 E\left[w(t)\right]^{-1} \nabla E\left[w(t)\right] \tag{5}$$

The update rule of the LM method can be written as follows:

$$w(t+1) = w(t) - \left(H - \eta I\right)^{-1} \nabla E\left[w(t)\right] \tag{6}$$

2.3 Input Variables

The primary aim of this paper is to determine the extent to which the GPU can reduce MLP training computational costs. Hence, to simplify the problem, only historical load data are considered for forecasting. The hourly load data of one year in the Korean market (1 Dec., 2007–30 Nov. 2008), including a total of 8784 observations, are used in the case studies. Four data sets, each containing three months of data are constructed. They represent the four different seasons and variations in demand over one year in Korea. The load data of one year are represented in Figs. 2-3 and Table 1.

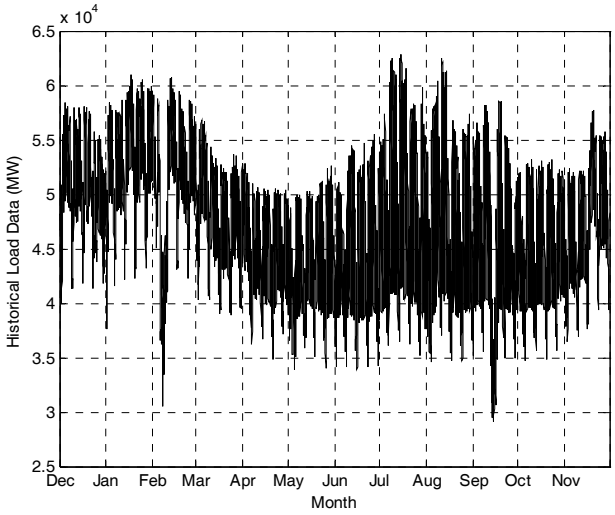


Fig. 2. Historical Load Data vs. Month.

Table 1. Load Data Analysis Results (MW)

Seasons	Mean	Max	Min
Winter (Dec. - Feb.)	51431.59	60947	30472
Spring (Mar. - May)	45972.07	56791	33884
Summer (Jun. - Aug.)	47413.35	62794	33880
Autumn (Sep. - Nov.)	46209.65	58510	29167

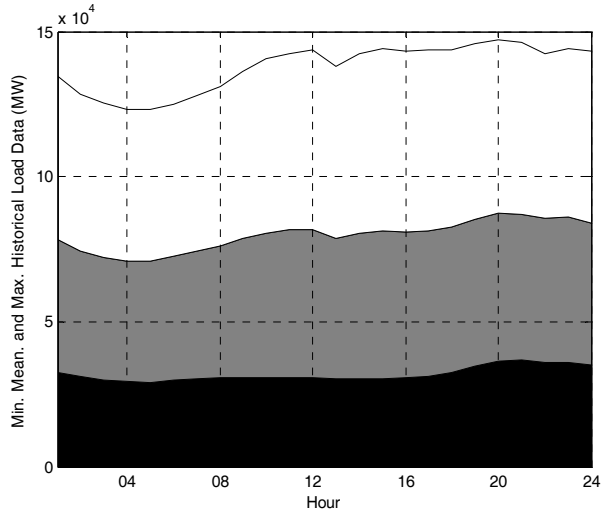


Fig. 3. Minimum, Mean, and Maximum Historical Load Data vs. Daily Hour.

As shown in the figures and table, significant differences exist among the different seasons. The highest load occurs in summer and the second highest occurs in winter. The loads in spring and autumn slight differ because temperature also changes with every season; summer has the highest temperature and winter has the lowest temperature. Hence, special attention should be given to differences among seasons using different neural networks. In this way, the training process becomes easier and the chances of obtaining better results are increased. We consider the four neural network models for the four seasons independently.

3. GPU Computing with CUDA

Recent computer graphics hardware have many vector processing units, as well as high memory bandwidths; hence, GPUs have evolved into powerful programmable processors, faster than CPUs in terms of the execution of parallel algorithms. Current trends in GPU design and configuration have given GPUs larger dedicated memories, higher bandwidth to graphics memories, and increased internal parallelism. The instructions of the program in shaders are close to assembly language because each has a direct hardware implementation. The new flexibility introduced by vertex shaders allows not only the naturalistic rendering of surfaces, but also brings GPUs closer to a general purpose parallel processor. In addition, current GPUs are designed with ever-increasing degrees of programmability, known as CUDA API [18]. Furthermore, for computationally-intensive applications, the data-parallel architecture of GPUs delivers dramatic performance gains compared to CPUs. In a number of instances, extensions to alternative graphics algorithms and scientific computing problems have been explored. It is the latter aspect that is exploited for better performance in MLP neural networks.

The idea of running neural networks on GPUs is to determine whether shader programs can run on parallel data processing systems. The proposed computing scheme is implemented on an Nvidia Tesla C1060 GPU with 240 1.3 GHz processor cores and 4 GB of GDDR3 memory at a bandwidth of 102 GB/s. The theoretical peak processing performance is up to 933 GFLOPS. Neural network algorithms generally apply primitives, such as inner products, outer products, linear algebra to vectors or matrices, nonlinearities (e.g., sigmoid and thresholding) applied to vectors or matrices, and matrix transposition.

As shown in Fig. 4, the hardware architecture of GPUs has high parallelism in terms of individual processor and memory allocation, favoring efficient streaming computation. Each grid computing unit also has an independent data bus to access global memory, an efficient way for operating large data sets synchronically. The benefits to linear algebraic calculations in particular have been demonstrated in many fields [19], [20]. Based on efficient matrix/vector calculation performance, neural networks are particularly well-suited for GPU implementation, provided that the weights reside on the GPU. Therefore, MLP neural networks can be implemented on the GPU to compute activation levels for neurons.

The flowchart is illustrated in Fig. 5. First, the CPU side workloads prepare initial training parameters and weights, and transmit the data set to the GPU via the graphics port bus. The data structures are then allocated in the GPU shared memory. After the vertex shader programs are defined in the GPU, the learning process can be activated. During the training process, the learning parameters can be transferred back to the CPU at long regular intervals to verify the learning progress without introducing overheads. Finally, when the training session is completed, after a fixed number of iterations, or when a desired error threshold

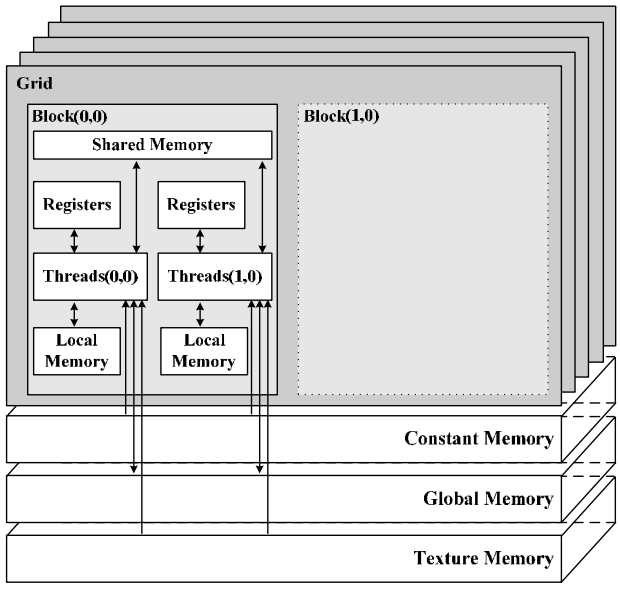


Fig. 4. GPU Structure with a Single Block.

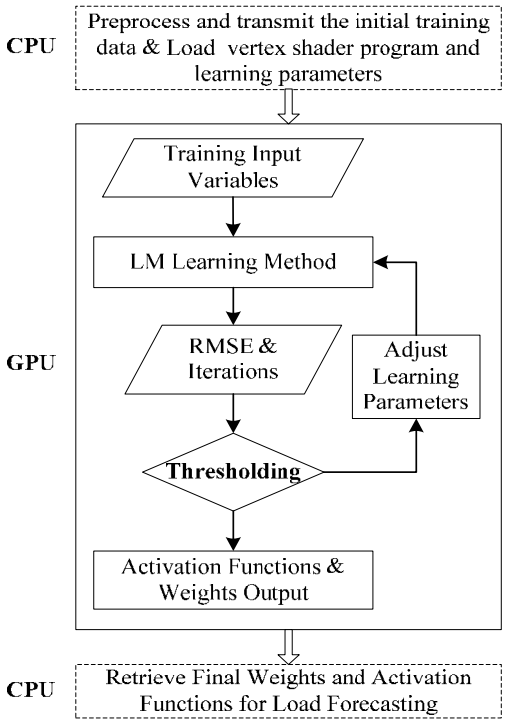


Fig. 5. Data Flow Diagram for MLP Training Algorithms on the GPU Structure.

has been achieved, the training process is terminated and the learning parameters are downloaded to the CPU and saved. However, GPUs have been observed to underperform either when significant overheads in calculations are incurred or when algorithms are not sufficiently parallel.

4. Demand Forecasting Case Study

In this section, the proposed forecasting method is tested using the Korean electricity market historical load demand series. The proposed method is used on the data series to illustrate its ability to handle demand data series. Our estimated sample loads consist of one year of daily observations. In one day, a total of 24 load demand data points must be predicted. As presented above, demand data are grouped into four seasons.

4.1 Preprocessing

Before data are used as input vectors to MLP, they may be subjected to some form of pre-processing, usually intended to make forecasting more accurate and manageable. Pre-processing of data is needed to remove outliers, missing values, or any irregularities because neural networks are sensitive to such defective data [21]. We use "standard demand" as input variables, which are represented as follows:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{7}$$

The load forecast from the neural forecaster is compared to the actual load data and the error is calculated. Two common criteria are used to evaluate forecast accuracy: mean absolute error (MAE) and mean absolute percentage error (MAPE) [11].

$$MAE = \frac{1}{N_h}\sum_{p=1}^{N_h}\left|y_p - \bar{y}_p\right| \tag{8}$$

$$MAPE = \frac{1}{N_h}\sum_{p=1}^{N_h}\frac{\left|y_p - \bar{y}_p\right|}{y_p}\times 100\% \tag{9}$$

where y_p is the actual load; $\bar{y}_p$ is the forecasted load; N_h is the sample number; and h is the hour index.

In the following section, the demand is forecasted according to four seasons.

Table 2. Load Data for Training and Forecasting (MW)

Seasons	Training	Forecasting	Total
Winter	1576 (1600-24)	560 (584-24)	2184
Spring	1576 (1600-24)	584 (608-24)	2208
Summer	1576 (1600-24)	584 (608-24)	2208
Autumn	1576 (1600-24)	560 (584-24)	2184

4.2 Forecasting Results

The results are based on data obtained from ten trials; only results from one case study are shown in the following figures and tables. Figs. 6a and 6b show load forecasting results in winter. The accuracy of the forecasted load curves is evident. The results give a MAPE value of 1.17%. As shown in Figs. 7a and 7b, the forecasted and actual load curves are also close to each other. Compared with the power demand in winter, the demand is reduced in spring.

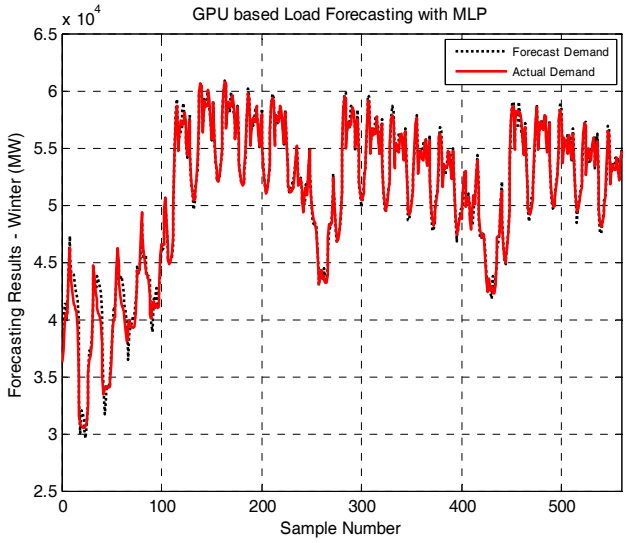


Fig. 6a. Demand Forecasting Results in Winter.

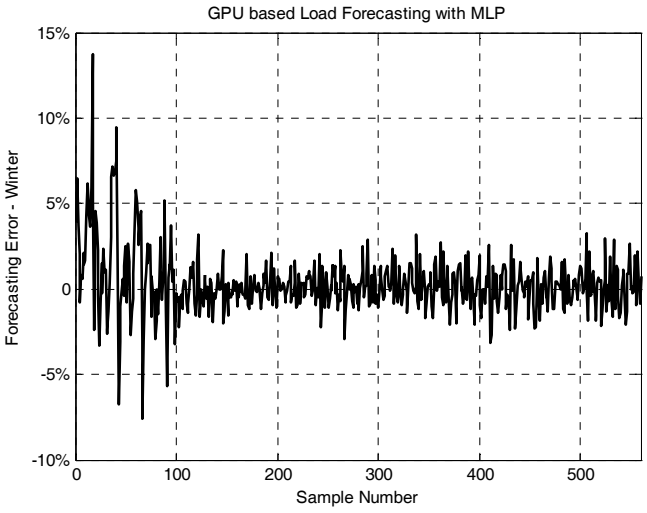


Fig. 6b. Demand Forecasting Error in Winter.

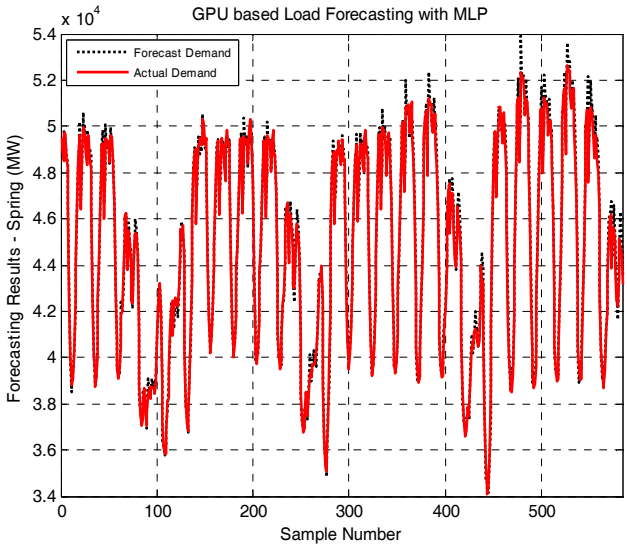


Fig. 7a. Demand Forecasting Results in Spring.

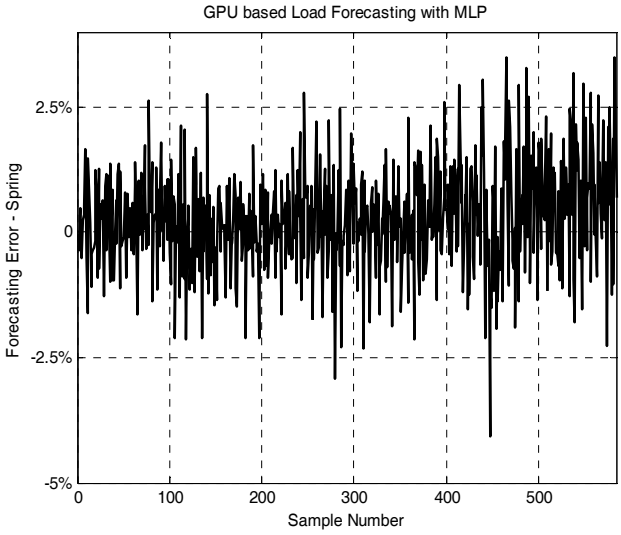


Fig. 7b. Demand Forecasting Error in Spring.

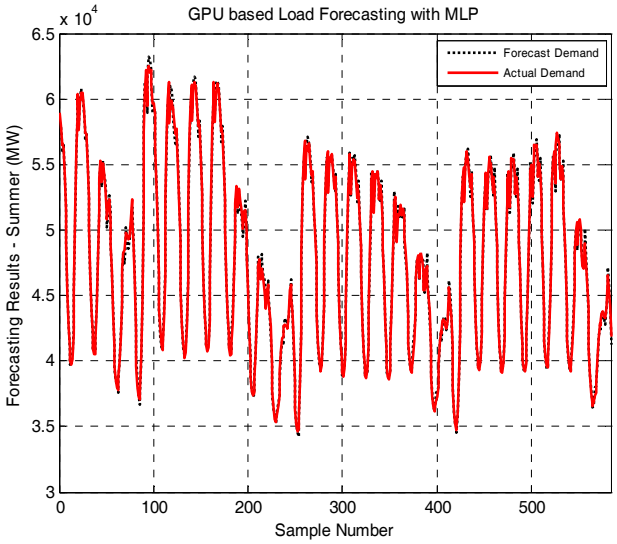


Fig. 8a. Demand Forecasting Results in Summer.

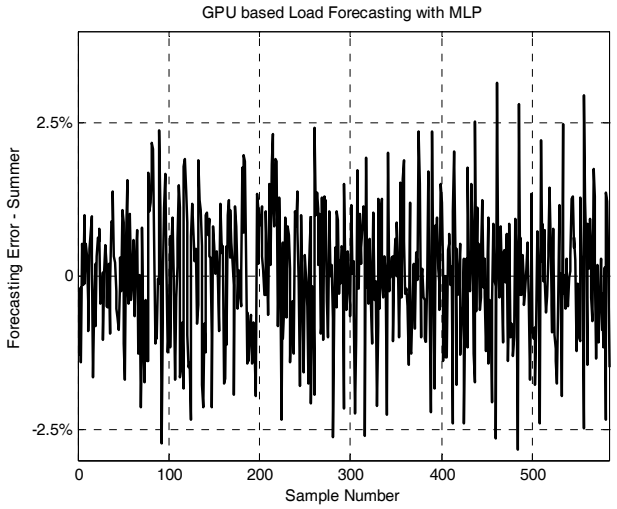


Fig. 8b. Demand Forecasting Error in Summer.

The spring MAPE value is 0.89%, which is comparatively lower than that of winter. As shown in Figs. 8a and 8b, the forecasted load curve almost coincides with the actual load curve during the summer period. In the three summer months, the temperature reaches the maximum, causing the increase in power demand. The forecasting errors are, however, fairly small and the value of MAPE is only 0.83%. As shown in Figs. 9a and 9b, the autumn forecast is quite accurate compared with the actual demand in autumn. The autumn power demand almost reaches 6000 MW, a little higher than in spring. With the proposed technique, the MAPE value is 1.15%.

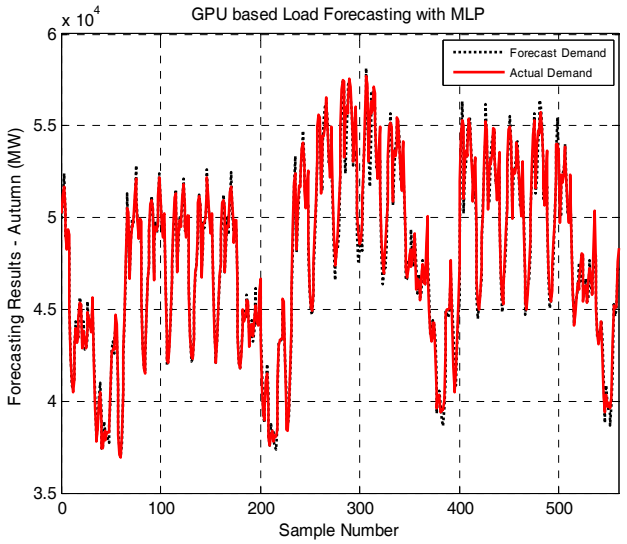


Fig. 9a. Demand Forecasting Results in Autumn.

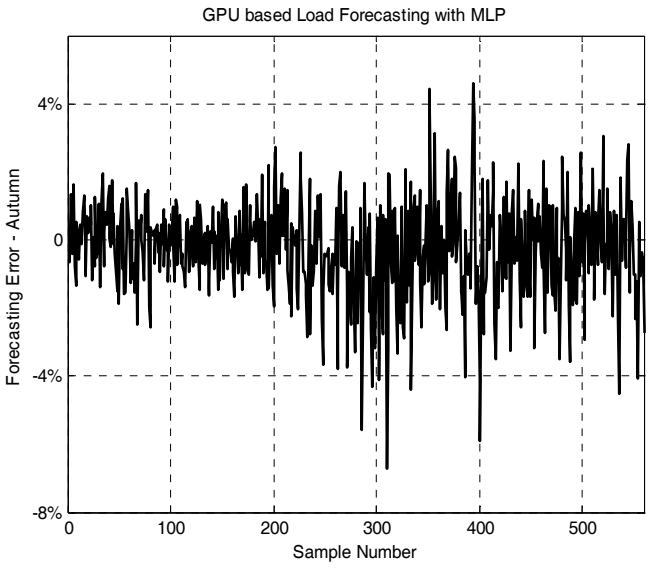


Fig. 9b. Demand Forecasting Error in Autumn.

4.3 Results Analysis

The average computational costs of the two methods are given in the following tables. All the CPU programs are run on a 2.66 GHz Intel Core 2 PC with 2 GB RAM.

Table 3. Comparison of results : CPU- and GPU-based MLP

Seasons	Average Computational Cost (s)		
	CPU	GPU	CPU/GPU
Winter	14.64	4.01	3.65
Spring	13.35	3.95	3.38
Summer	11.53	3.81	3.02
Autumn	16.32	4.57	3.57
Overall	13.96	4.09	3.41

Table 4. Load Data Forecasting Results

Seasons	MAE	MAPE
Winter	549.99	1.17%
Spring	396.76	0.89%
Summer	405.26	0.83%
Autumn	558.26	1.15%
Overall	477.57	1.01%

As shown in Tables 3 and 4, the GPU-based MLP greatly reduces the learning time compared with the CPU-based MLP. To simplify the problem, only historical power demand is considered as input variable. In practical applications however, other variables that influence the loads must be considered as input vectors. The GPU version is about 9–11 seconds faster than the CPU version (Table III). Through other experiments, we also find that if the code is optimized and simplified, the CPU to GPU speed ratio can reach 4.

The strength of GPUs is in streaming computation and synchronous processing. This is the basis of significant improvements in computational efficiency provided by GPUs. If the code is strictly programmed according to the instructions and we do not frequently access data between CPUs and GPUs, the advantage of GPUs can be very obvious. If more input variables are used, the overall efficiency is increased.

5. Conclusion

The acceleration of the neural network learning task involving numerous data-parallel computations is promising. The evaluation results demonstrate the great potential of our prototype for massive neural network learning tasks. The graphics hardware can efficiently work, especially if the task is implemented using data-parallel instructions supported by hardware. The simulation results show that this approach is much faster than CPU-based parallel computing. The computational cost of GPU-based computing is only one-third that of CPU-based computing, demonstrating the great potential of the GPU-based demand forecast process. It also opens new possibilities for much higher computational efficiency for other real-time power system data analyses.

References

[1] P.R.J. Campbell, K. Adamson, "Methodologies for load forecasting," *Int. IEEE Conf. Intell. Syst.*, pp. 800-806, Sept. 2006.

[2] L. Xu, Z.Y. Dong, and A. Tay, "Time series forecast with Elman neural networks and genetic algorithms," in K.C. Tan, M.H. Lim, X. Yao, and L.P. Wang (ed): *Recent Advances in Simulated Evolution and Learning*, World Scientific series, Vol. 2, Advances in Natural Computation, 2004.

[3] Z. Xu, Z.Y. Dong, and W. Liu, "Neural network models for electricity market forecasting," in D. Wang and N.K. Lee (ed): *Neural Networks Applications in Information Technology and Web Engineering*, Borneo Publications, 2005.

[4] Z.Y. Dong, "A neural network based framework for electricity load and price forecasting," *Int. Conf. Comput. Intell., Robot. and Auton. Syst.*, Singapore, pp. 88-93, Nov. 2001.

[5] B.L. Zhang, Z.Y. Dong, "An adaptive neural-wavelet model for short term load forecasting," *Int. J. Electr. Power Syst. Res.*, Vol. 59, No. 2, pp. 121-129, Sep. 2001.

[6] H.A. Salama, A.F.A. El-Gawad, H.M. Mahmoud, E.A. Mohamed, S.M. Saker, "Short-term load forecasting investigations of Egyptian electrical network using ANNs," *Int. Uni. Power Eng. Conf.*, Vol. 4-6, pp. 550-555, Sep. 2007.

[7] G.C. Liao, "A novel particle swarm optimization approach combined with fuzzy neural networks for short-term load forecasting," *IEEE Power Engineering Society General Meeting*, Jun. 2007.

[8] Y. Xu, Z.Y. Dong, K. Meng, J.H. Zhao, and H.M. Yang "A short-term load forecasting model for Australian NEM," *Chin. Univ. Conf. Power Syst. Automat.*, Changsha, Oct. 2009.

[9] K. Liu, S. Subbarayan, R.R. Shoults, M.T. Manry, C. Kwan, F.I. Lewis, J. Naccarino, "Comparison of very short-term load forecasting techniques," *IEEE Trans. Power Syst.*, Vol. 11, No. 2, pp. 877-882, May 1996.

[10] K. Meng, Z.Y. Dong, and K.P. Wong, "Self-adaptive RBF neural network for short-term electricity price forecasting," *IET Gen Trans & Dist.*, Vol. 3, No. 4, pp 325-335,Apr. 2009.

[11] K. Meng, Z.Y. Dong, H.G. Wang, and Y.Y. Wang, "Comparisons of machine leaning methods for electricity reference price forecasting," *ISNN 2009: Int. Symp. Neural Netw.*, Wuhan, pp. 827-835, May 2009.

[12] X. Yin, Z.Y. Dong, and P. Zhang, "Fundamentals of emerging techniques," in Z.Y. Dong, P. Zhang, et al. (edit): *Emerging Techniques in Power System Analysis*, Springer, pp. 23-44, Jan. 2010.

[13] T. Senjyu, H. Takara, K. Uezato, T. Funabashi, "One-hour-ahead load forecasting using neural networks," *IEEE Trans. Power Syst.*, Vol. 17, No. 1, pp. 113-118, Feb. 2002.

[14] C. Lursinsap, J.H. Kim, "Parallel learning for back-propagation network in binary field," *IEEE Int. Symp. Circuits Syst.*, Vol. 3, pp. 1477-1480, Jun. 1991.

[15] H. Schabauer, E. Schikuta, T. Weishaupl, "Solving very large traveling salesman problems by SOM parallelization on cluster architectures," *Int. Conf. Parallel and Distri. Comput., Appl. and Technol.*, pp. 954-958, Dec. 2005.

[16] I. Buck, "GPU computing: programming a massively parallel processor," *Int. Symp. Code Gen. and Optimiz.*, pp. 17, Mar. 2007.

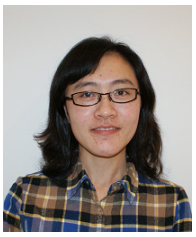
[17] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Q. J. Appl. Math.*, Vol. II, No. 2, pp. 164-168, 1944.

[18] NVIDIA CUDA Compute Unified Device Architecture, Programming Guide Version 2.1 Beta, Dec. 2008.

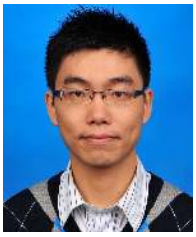
[19] J. Bolz, I. Farmer, E. Grinspun, and P. Schröder, "Sparse matrix solvers on the GPU: conjugate gradients and multigrid," *ACM Trans. Graph.*, Vol. 22, No. 3, pp. 917-924, Aug. 2003.

[20] I. Buck, T. Foley, D. Horn, J. Sugerman, K. Fatahalian, M. Houston, and P. Hanrahan, "Brook for GPUs: stream computing on graphics hardware," *ACM Trans. Graph.*, Vol. 23, No. 3, pp. 777-786, Aug. 2004.

[21] H.S. Hippert, C.E. Pedreira, R.C. Souza, "Neural networks for short-term load forecasting: a review and evaluation," *IEEE Trans. Power Syst.*, Vol. 16, No. 1, pp. 44-55, Feb. 2001.



Ting He received B.Eng and M.Eng degrees from Southeast University, China, in 2003 and 2006 respectively. She is currently a PhD candidate in The University of Queensland, Australia. Her research interests include investment strategy valuations of wind power.



Ke Meng received the Ph.D. from The University of Queensland, Australia, in 2009. He is now a research fellow at the Department of Electrical Engineering, The Hong Kong Polytechnic University. His research interest includes wind power, power system stability analysis and control, power market analysis, data mining and intelligent algorithms.

ZhaoYang Dong obtained his PhD from The University of Sydney, Australia in 1999. He is currently an associate professor at The Hong Kong Polytechnic University, Hong Kong. He previously held academic positions with The University of Queensland, Australia. He also held industrial positions with Transend Networks, Tasmania, Australia. His research interest includes power system planning, power system security, stability and control, load modeling, electricity market, and computational intelligence and its application in power engineering.

Yong-Taek Oh received his B.S degree from the Sungsil University, Korea in 1980 and the M.Sc and Ph.D degree Electrical Engineering from the Yonsei University, Korea in 1982 and 1987 respectively in Electrical Engineering. From 1987 to 1991, He was with the Computer Center of Korea Electric Power Corporation as a Section Chief. He joined the faculty of Korea University of Technology and education, Cheonan, Korea in 1991 where he is currently a professor in the school of Information Technology. His areas of research interest are power system protection and control, power system stability, fault analysis and computational intelligence and its application in power engineering.

Yan Xu received the B.Eng. from South China University of Technology, in 2008. He is now a Ph.D. candidate at Department of Electrical Engineering, The Hong Kong Polytechnic University. His current research interests include power system security assessment, wind power, data mining, computational intelligence and their application in power engineering.