Fast Information Retrieval from Big Data by using Neural Networks Implemented in the Frequency Domain

Hazem M. El-Bakry, Nikos E. Mastorakis, and Michael E. Fafalios

***Abstract*—**The objective of storing data is to retrieve it as requested in a fast way. In this paper, a new efficient model for fast retrieving of specific information from big data is presented. Fast neural networks are used to find the best matching between words in query and stored big data. The idea is to accelerate the searching operation in a big data. This is done by applying cross correlation between the given query and the big data in the frequency domain rather than time domain. Furthermore, neural networks are used to retrieve information from big data even these data are noised or distorted. The mathematical prove for the acceleration process is analyzed and a formula for the theoretical speed up ratio is given. Simulation results confirm the theoretical considerations.

Keywords— Big data, Cross Correlation, Frequency Domain, Neural networks.

I. Introduction

Advances in digital sensors, communications, computation, and storage have created huge collections of data, capturing information of value to business, science, government, and society. Big data is term that describes the exponential growth of vast amount of data, both structured and unstructured. It is the ocean of information we swim in every day. Examples are the data flow in our computers, mobile devices, and machine sensors [1-6].

Big data usually includes data sets with sizes beyond the ability of commonly used software tools to capture, curate, manage, and process the data within a tolerable elapsed time. Big data sizes are a constantly moving target, as of 2012 ranging from a few dozen terabytes to many petabytes of data in a single data set. The target moves due to constant improvement in traditional DBMS technology as well as new databases like NoSQL and their ability to handle larger amounts of data. With this difficulty, new platforms of "big data" tools are being developed to handle various aspects of large quantities of data. Examples of Big Data include Big Science, sensor networks, social networks, big social data analysis, Internet documents, Internet search indexing, call detail records, astronomy, atmospheric science, bio geochemical, biological, and other complex and often interdisciplinary scientific research, military surveillance, forecasting drive times for new home buyers, medical records, photography archives, video archives, and large-scale e-commerce. When dealing with big data, organizations face difficulties in being able to create, manipulate, and manage theses big data. Analyzing big data is a complex problem in business processing because standard tools and procedures are not designed to search and analyze massive datasets [10-22].

Therefore, there is a need for a strategy to enhance big data processing to improve policy and service delivery. The main objective of this paper is to manipulate massive volumes of both structured and unstructured data. Such volumes are so large that it's difficult to process using traditional database and software techniques. The idea of applying cross correlation between input patterns and stored information was applied successfully in many different applications by using neural networks [7-9]. Here, we make use of these previous results to increase the speed of information retrieval from big data. Neural Networks are used to detect the required information from big data even these data are noised or distorted.

The rest of this paper is organized as follows. The theory of fast information retrieval in the frequency domain is presented in section II. Simulation Results are given in section III. Finally conclusions are given.

II. Fast Information Retrieval by using Cross Correlation Implemented in the Frequency Domain

Information processing by using neural networks is divided into two parts. First neural networks are trained to recognize the input patterns. In the test phase, each position in the incoming matrix is processed and tested for the required data (code) by using neural networks. At each position in the input one dimensional matrix, each sub-matrix is multiplied by a window of weights, which has the same size as the sub-matrix. The outputs of neurons in the hidden layer are multiplied by the weights of the output layer. Thus, we may conclude that the whole problem is a cross correlation between the incoming serial data and the weights of neurons in the hidden layer. The

H. M. El-Bakry is assistant professor with Dept. of Information Systems - Faculty of Computer Science and Information Systems – Mansoura University – Egypt. (phone: +2-050-2349340, fax: +2-050-2221442, e-mail: helbakry5@yahoo.com).

N. E. Mastorakis, is with the Technical University of Sofia, BULGARIA.

M. E. Fafalios is with the Hellenic Naval Academy (ASEI), Piraeus, Greece Sector of Electronics and Communications, Piraeus, Greece (phone 0030 210 4581644, Fax 0030210 4181768, fafalios@hna.gr)

convolution theorem in mathematical analysis says that a convolution of f with h is identical to the result of the following steps: let F and H be the results of the Fourier Transformation of f and h in the frequency domain. Multiply F and H* in the frequency domain point by point and then transform this product into the spatial domain via the inverse Fourier Transform. As a result, these cross correlations can be represented by a product in the frequency domain. Thus, by using cross correlation in the frequency domain, speed up in an order of magnitude can be achieved during the test phase [7-9]. Assume that the size of the input pattern is 1xn. In the test phase, a sub matrix I of size 1xn (sliding window) is extracted from the tested matrix, which has a size of 1xN. Such sub matrix, which contains the input pattern, is fed to the neural network. Let W_i be the matrix of weights between the input sub-matrix and the hidden layer. This vector has a size of 1xn and can be represented as 1xn matrix. The output of hidden neurons h(i) can be calculated as follows [7-9]:

$$h_i = g\left(\sum_{k=1}^{n} W_i(k) I(k) + b_i \right) \quad (1)$$

where g is the activation function and b(i) is the bias of each hidden neuron (i). Equation 1 represents the output of each hidden neuron for a particular sub-matrix I. It can be obtained to the whole input matrix Z as follows [7-9]:

$$h_i(u) = g\left(\sum_{k=-n/2}^{n/2} W_i(k)\ Z(u+k) + b_i \right) \quad (2)$$

Eq.1 represents a cross correlation operation. Given any two functions f and d, their cross correlation can be obtained by [23]:

$$d(x) \otimes f(x) = \left(\sum_{n=-\infty}^{\infty} f(x+n) d(n) \right) \quad (3)$$

Therefore, Eq. 2 may be written as follows [7-9]:

$$h_i = g\left(W_i \otimes Z + b_i \right) \quad (4)$$

where h_i is the output of the hidden neuron (i) and $h_i(u)$ is the activity of the hidden unit (i) when the sliding window is located at position (u) and (u) $\in$ [N-n+1].

Now, the above cross correlation can be expressed in terms of one dimensional Fast Fourier Transform as follows [7-9]:

$$W_i \otimes Z = F^{-1}\left(F(Z) \bullet F^*\left(W_i \right) \right) \quad (5)$$

Hence, by evaluating this cross correlation, a speed up ratio can be obtained comparable to traditional neural networks. Also, the final output of the neural network can be evaluated as follows:

$$O(u) = g\left(\sum_{i=1}^{q} W_o(i)\, h_i(u) + b_o \right) \quad (6)$$

where q is the number of neurons in the hidden layer. O(u) is the output of the neural network when the sliding window located at the position (u) in the input matrix Z. W_o is the weight matrix between hidden and output layer.

The complexity of cross correlation in the frequency domain can be analyzed as follows:

1- For a tested matrix of 1xN elements, the 1D-FFT requires a number equal to $N\log_2 N$ of complex computation steps [24]. Also, the same number of complex computation steps is required for computing the 1D-FFT of the weight matrix at each neuron in the hidden layer.

2- At each neuron in the hidden layer, the inverse 1D-FFT is computed. Therefore, q backward and (1+q) forward transforms have to be computed. Therefore, for a given matrix under test, the total number of operations required to compute the 1D-FFT is $(2q+1)\ N\log_2 N$.

3- The number of computation steps required by fast neural networks (FNNs) is complex and must be converted into a real version. It is known that, the one dimensional Fast Fourier Transform requires $(N/2)\log_2 N$ complex multiplications and $N\log_2 N$ complex additions [24]. Every complex multiplication is realized by six real floating point operations and every complex addition is implemented by two real floating point operations. Therefore, the total number of computation steps required to obtain the 1D-FFT of a 1xN matrix is:

$$\rho = 6((N/2)\log_2 N) + 2(N\log_2 N) \quad (7)$$

which may be simplified to:

$$\rho = 5N\log_2 N \quad (8)$$

4- Both the input and the weight matrices should be dot multiplied in the frequency domain. Thus, a number of complex computation steps equal to qN should be considered. This means 6qN real operations will be added to the number of computation steps required by FNNs.

5- In order to perform cross correlation in the frequency domain, the weight matrix must be extended to have the same size as the input matrix. So, a number of zeros = (N-n) must be added to the weight matrix. This requires a total real number of computation steps = q(N-n) for all neurons. Moreover, after computing the FFT for the weight matrix, the conjugate of this matrix must be obtained. As a result, a real number of computation steps = qN should be added in order to obtain the conjugate of the weight matrix for all neurons. Also, a number of real computation steps equal to N is required to create butterflies complex numbers ($e^{-jk(2\Pi n/N)}$), where 0<K<L. These (N/2) complex numbers are multiplied by the elements of the

input matrix or by previous complex numbers during the computation of FFT. To create a complex number requires two real floating point operations. Thus, the total number of computation steps required for FNNs becomes:

$$\sigma=(2q+1)(5N\log_2 N)+6qN+q(N-n)+qN+N \quad (9)$$

which can be reformulated as:

$$\sigma=(2q+1)(5N\log_2 N)+q(8N-n)+N \quad (10)$$

6- Using sliding window of size 1xn for the same matrix of 1xN pixels, q(2n-1)(N-n+1) computation steps are required when using traditional neural networks (TNNs) to process (n) input data. The theoretical speed up factor η can be evaluated as follows:

$$\eta = \frac{q(2n-1)(N-n+1)}{(2q+1)(5N\log_2 N)+q(8N-n)+N} \quad (11)$$

III. Simulation Results

TNNs accept serial input data with fixed size (n). Therefore, the number of input neurons equals to (n). Instead of treating (n) inputs, the proposed new approach is to collect all the incoming data together in a long vector (for example 100xn). Then the input data is tested by time delay neural networks as a single pattern with length L (L=100xn). Such a test is performed in the frequency domain as described before.

Eq. 11 is also true for recurrent neural networks. The theoretical speed up ratio for processing short successive (n) data in a long input vector (L) using recurrent neural networks is listed in tables 1, 2, 3, 4, 5 and 6. Also, the practical speed up ratio for manipulating matrices of different sizes (L) and different sized weight matrices (n) using a 2.7 GHz processor and MATLAB is shown in table 4. An interesting point is that the memory capacity is reduced when using FNNs. This is because the number of variables is reduced compared to TNNs.

Another point of interest should be noted. In TNNs, if the whole input data (N) is available, then there is a waiting time for each group of (n) input data so that conventional neural networks can release their output for the previous group of (n) data. In contrast, FNNs can process the total N data directly with zero waiting time. For example, if the total (N) input data is appeared at the input neurons, then:

1- TNNs can process only data of size (n) as the number of input neurons = (n).
2- The first group of (n) data is processed by TNNs.
3- The second group of (n) data must wait for a waiting time = τ, where τ is the response time consumed by TNNs for treating each group of (n) input data.
4- The third group of (n) data must wait for a waiting time = 2τ corresponding to the total waiting time required by TNNs for treating the previous two groups.
5- The fourth (n) data must wait for a waiting time = 3τ.
6- The last group of (n) data must wait for a waiting time = (N-n)τ.

As a result, the wasted waiting time in the case of TNNs is (N-n)τ. In the case of FNNs, there is no waiting time as the whole input data (Z) of length (N) will be processed directly and the time consumed is the only time required by FNNs itself to produce their output.

IV. Conclusion

A new approach for testing big data with neural networks has been presented. The operation of neural networks during the test phase has been accelerated. This has been done by applying cross correlation between the whole input patterns and the input weights of neural networks in the frequency domain rather than time domain. The mathematical proof for the acceleration process has been introduced. A formula for the theoretical speed up ratio has been given. Simulation results have confirmed the theoretical considerations.

References

[1] S. del Rio, V. L pez, J.M. Benitez, F. Herrera, On the use of MapReduce for imbalanced big data using Random Forest, ElSEVIER, (2014).
[2] X.-W. Chen, Big Data Deep Learning: Challenges and Perspectives, IEEE, (2014).
[3] Yingyi Bu, Vinayak Borkar, Michael J. Carey, Joshua Rosen, Neoklis Polyzotis, Tyson Condie, Markus Weimer, Raghu Ramakrishnan, Scaling Datalog for Machine Learning on Big Data, (2012).
[4] A. Cassioli , A. Chiavaioli , C. Manes , M. Sciandrone, An incremental least squares algorithm for large scale linear classification, ElSEVIER, (2012).
[5] C.L. Philip Chen, Chun-Yang Zhang, Data-intensive applications, challenges, techniques and technologies: A survey on Big Data, ElSEVIER, (2013).
[6] Alicia Fernández a, ÁlvaroGómez a, FedericoLecumberry a,n, ÁlvaroPardo b, Ignacio Ramírez a, Pattern Recognitionin Latin Americainthe "Big Data" Era, ElSEVIER,(2014).
[7] Hazem M. El-Bakry, Nikos E. Mastorakis, Michael E. Fafalios, "Fast Information Retrieval from Big Data by using Cross Correlation in the Frequency Domain," Proc. of IEEE IJCNN 2013, Dallas Tx, USA, August 4-9, 2013, pp. 366-272.
[8] Hazem M. El-Bakry, "An Efficient Algorithm for Pattern Detection using Combined Classifiers and Data Fusion," Information Fusion Journal, vol. 11, 2010, pp. 133-148.
[9] Hazem M. El-Bakry, "Fast Virus Detection by using High Speed Time Delay Neural Networks," Journal of Computer Virology, vol. 6, no. 2, 2010, pp. 115-122.
[10] http://www.storageswitzerland.com/Articles/Entries/2011/6/16_Designing_Big_Data_Storage_Infrastructures.html
[11] "Switched Fabric" (Wikipedia), http://en.wikipedia.org/wiki/Switched_fabric, accessed May 27, 2014.
[12] "Big Data" (Wikipedia), http://en.wikipedia.org/wiki/Big_data
[13] "big data" http://www.webopedia.com/TERM/B/big_data.html
[14] "Big Data" Industry Report 2014 IMEXResearch.com
[15] "NextGen Infrastructure for Big Data 2012 Storage Networking Industry Associa on", Joseph White, Anil Vasuedeva "10 emerging technologies for Big Data" interviewed Dr. Satwant Kaur about the 10 emerging technologies that will drive Big Data forward December 4, 2012 .
[16] "Leveraging Hadoop-Based Big Data Architectures for a Scalable, High-Performance Analytics Platform" 2000488-001-EN Sept 2012 2012 Juniper Networks.
[17] "Making the Most of Big Data" Dr. Hossein Eslambolchi, former CTO of AT&T, is chairman and CEO of 2020 Venture Partners.
[18] "Big Security for Big Data: Addressing Security Challenges for the Big Data Infrastructure", by Y.Demchenko, P.Membrey, C.Ngo, C. de Laat,

D.Gordijenko Submitted to Secure Data Management (SDM'13) Workshop. Part of VLDB2013 conference, 26-30 August 213, Trento, Italy
[19] "Big Data: What It Means for Data Center Infrastructure", Krishna Kallakuri owner and vice president of DataFactZ SEPTEMBER 5, 2013
[20] "Big Data, Big Service" 7th July 2013 by Tony Shan
"Big Data Solution Offering". MIKE2.0. Retrieved 14 May 2014.
[21] "Big Data Definition". MIKE2.0. Retrieved 9 March 2013.
[22] Boja, C; Pocovnicu, A; Bătăgan, L. (2012). "Distributed Parallel Architecture for Big Data".Informatica Economica 16 (2): 116–127.
[23] Klette R., and Zamperon, "Handbook of image processing operators, " John Wiley & Sonsltd, 1996.
[24] Cooley, J. W. and Tukey, J. W., "An algorithm for the machine calculation of complex Fourier series," Math. Comput. 19, 1965, pp. 297–301

Hazem M. El-Bakry (Mansoura, EGYPT 20-9-1970) received B.Sc. degree in Electronics Engineering, and M.Sc. in Electrical Communication Engineering from the Faculty of Engineering, Mansoura University - Egypt, in 1992 and 1995 respectively. Dr. El-Bakry received Ph. D degree from University of Aizu - Japan in 2007. Currently, he is associate professor at the Faculty of Computer Science and Information Systems – Mansoura University – Egypt. His research interests include neural networks, pattern recognition, image processing, biometrics, cooperative intelligent systems and electronic circuits. In these areas, he has published many papers in major international journals and refereed international conferences. According to academic measurements, now the total number of citations for his publications is 1972. The H-index of his publications is 24 and G-index is 19. Dr. El-Bakry has the United States Patent No. 20060098887, 2006. Furthermore, he is associate editor for journal of computer science and network security (IJCSNS) and journal of convergence in information technology (JCIT). In addition, is a referee for IEEE Transactions on Signal Processing, Journal of Applied Soft Computing, the International Journal of Machine Graphics & Vision, the International Journal of Computer Science and Network Security, Enformatika Journals, WSEAS Journals and many different international conferences organized by IEEE. Moreover, he has been awarded the Japanese Computer & Communication prize in April 2006 and the best paper prize in two conferences cited by ACM. He has also been awarded Mansoura university prize for scientific publication in 2010 and 2011. Dr. El-Bakry has been selected in who Asia 2006 and BIC 100 educators in Africa 2008.

Nikos E. Mastorakis is full professor with the Technical University of Sofia, BULGARIA.

.

M. E. Fafalios is with the Hellenic Naval Academy (ASEI), Piraeus, Greece Sector of Electronics and Communications, Piraeus, Greece (phone 0030 210 4581644, Fax 0030210 4181768, fafalios@hna.gr)

Table 1. The theoretical speed up ratio (n=400).

Length of input data	Number of computation steps required for TNNs	Number of computation steps required for FNNs	Speed up ratio
10000	2.3014e+008	4.2926e+007	5.3613
40000	0.9493e+009	1.9614e+008	4.8397
90000	2.1478e+009	4.7344e+008	4.5365
160000	3.8257e+009	8.8219e+008	4.3366
250000	5.9830e+009	1.4275e+009	4.1912
360000	8.6195e+009	2.1134e+009	4.0786
490000	1.1735e+010	2.9430e+009	3.9876
640000	1.5331e+010	3.9192e+009	3.9119

Table 2. The theoretical speed up ratio (n =625).

Length of input data	Number of computation steps required for TNNs	Number of computation steps required for FNNs	Speed up ratio
10000	3.5132e+008	4.2919e+007	8.1857
40000	1.4754e+009	1.9613e+008	7.5226
90000	3.3489e+009	4.7343e+008	7.0737
160000	0.5972e+010	8.8218e+008	6.7694
250000	0.9344e+010	1.4275e+009	6.5458
360000	1.3466e+010	2.1134e+009	6.3717
490000	1.8337e+010	2.9430e+009	6.2306
640000	2.3958e+010	3.9192e+009	6.1129

Table 3. The theoretical speed up ratio (n =900).

Length of input data	Number of computation steps required for TRNNs	Number of computation steps required for FNNs	Speed up ratio
10000	4.9115e+008	4.2911e+007	11.4467
40000	2.1103e+009	1.9612e+008	10.7600
90000	4.8088e+009	4.7343e+008	10.1575
160000	0.8587e+010	8.8217e+008	9.7336
250000	1.3444e+010	1.4275e+009	9.4178
360000	1.9381e+010	2.1134e+009	9.1705
490000	2.6397e+010	2.9430e+009	8.9693
640000	3.4493e+010	3.9192e+009	8.8009

Table 4. The theoretical speed up ratio (n =1600).

Length of input data	Number of computation steps required for TRNNs	Number of computation steps required for FNNs	Speed up ratio
10000	6.5894e+008	4.2810e+007	15.3921
40000	2.8691e+009	1.9612e+008	14.6289
90000	6.6330e+009	4.7343e+008	14.0169
160000	1.1993e+010	8.8217e+008	13.5954
250000	1.8709e+010	1.4275e+009	13.1059
360000	2.6690e+010	2.1134e+009	12.6291
490000	3.6609e+010	2.9430e+009	12.4393
640000	4.7943e+010	3.9192e+009	12.2329

Table 5. The theoretical speed up ratio (n =2500).

Length of input data	Number of computation steps required for TRNNs	Number of computation steps required for FNNs	Speed up ratio
10000	8.3251e+008	4.2810e+007	19.4467
40000	3.8753e+009	1.9612e+008	19.7600
90000	9.0698e+009	4.7343e+008	19.1575
160000	1.6526e+010	8.8217e+008	18.7336
250000	2.6291e+010	1.4275e+009	18.4178
360000	3.8401e+010	2.1134e+009	18.1705
490000	5.2884e+010	2.9430e+009	17.9693
640000	6.9765e+010	3.9192e+009	17.8009

Table 6. Practical speed up ratio.

Length of input data	Speed up ratio (n=400)	Speed up ratio (n=625)	Speed up ratio (n=900)	Speed up ratio (n=1600)	Speed up ratio (n=2500)
10000	8.94	12.97	17.61	24.87	32.55
40000	8.60	12.56	17.22	24.42	32.43
90000	8.33	12.28	16.80	23.76	32.32
160000	8.07	12.07	16.53	23.34	32.23
250000	7.95	11.92	16.30	23.13	32.14
360000	7.79	11.62	16.14	22.96	32.07
490000	7.64	11.44	16.00	22.90	31.99
640000	7.04	11.27	15.89	22.88	31.96