JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Deep Learning Approach for Enhancing Fault Tolerance for Reliable Distributed System

Lokendra Gour[1], and Akhilesh A. Waoo[2]

[1]Department of Computer Science and IT, Faculty of Computer Applications, Information Technology and Sciences, AKS University, Satna (M.P.) 485001
E-mail: lokendra.gaur@gmail.com, akhileshwaoo@gmail.com

Abstract

Data in distributed system is dispersed in a structured, unstructured or semi-structured format. Data is distributed across various institutions organizations or individuals. These institutions are analogous to the working nodes. This dispersed data needs to be handled properly in two respects one for ensuring data security and privacy another is enhancing fault tolerance. Data security and privacy has become a prime concern for data centric applications. Robust fault tolerant platform is required for smooth functioning of the distributed system. The intuitive and robust model FedLearning is framed for distributed learning. FedLearning is based on ensemble learning, in which neural network models are deployed independently on each data unit at the local working node. All the local working model's parameters are combined and collected by the secure coordinating nodes. Federated learning performs the coordination among the working nodes. Coordination prevents the failures of individual working machines.

Keywords: Distributed System, Fault Tolerance, Federated Learning, Working Node

1. Introduction

Fault tolerant platform is an essential part of distributed computing for preventing the system from failures. Federated learning is a distributed and decentralized paradigm of protocols [1], [2], [3]. Federated learning approach is well suited for a distributed system because a set of worker machine (or client) can train the local models. Different chunks of datasets are distributed among the worker nodes or third parties. Here sections of dataset are not shared by the working computational nodes. Thus federated learning is also the most significant model for achieving data privacy and data security in addition to fault tolerance [4], [5]. We deployed a framework by using federated learning algorithms with a parameter server. The existing federated learning (FL) approaches highlights optimizing only one dimension of the target space. The proposed novel methods can reduce the communication cost and improve the efficiency of the distributed computing. The FedLearning method minimizes the adversarial effect with high convergence rate. This approach utilizes a straggler-aware, weighted aggregation for accuracy improvement [6], [7], [8]. FedLearning is capable to detect and diagnose the faults that occur frequently on end-user devices as well as on the edge. FedLearning is a novel communication efficient FL approach. It incorporates both the synchronous and asynchronous arrangements.

Federated learning is a multi-modal machine learning system that trains the algorithm among various distributed and decentralized edge devices that holds local datasets. The intelligent device such as PDAs, smart-phones, and desktops or tablets system has been scaling rapidly in recent years. Most of these devices are equipped with multiple sensors that allow them to produce and consume a huge amount of information. Distributed computing hierarchy consists of cloud, edge, and end-user devices [9], [10], [11], [12]. End-user devices train the local models and use local datasets. The parameter server is deployed at the cloud layer and performs the task of aggregating the local updates and upgrading the global model after receiving the updated local models. Edge works as an intermediate layer between the cloud and end-devices. Edge acts almost similar to the cloud, it performs the task of taking the output of end devices as input, applies aggregation and classification if necessary, and finally transfers its intermediate output to the cloud system for further processing if required [13], [14], [15].

The proposed algorithms is based on ensemble learning, in which local gradients obtained from working nodes are aggregated by separate module called aggregator. Stochastic Gradient Descent (SGD) algorithm is the most prominent machine learning algorithm. In this research work, the variant of SGD that is distributed SGD (DSGD) is proposed and it is deployed in the aforementioned model. DSGD is implemented as federated learning.

2. System Architecture

As illustrated in Figure 1 the proposed model is administered by a main controller known as central server or parameter server which is assisted by secondary controller. The primary function of main controller is to receive request from edge devices and then assign them to the appropriate machine for further processing. The key objective of central server is the orchestration of the various working modules [16], [17], [18], [19], [20]. The different modules and functions of the proposed architecture are organized and controlled by algorithmic modules. The key objective of the proposed model is to improve the fault tolerance by enhancing fault prediction module.

System topology for federated with distributed learning consists of a graph G = (V, E), where the V = set of working nodes (or sequential processes) and E = set of edges (bi/unidirectional communication channel or links) [21], [22],. Fig. 1 shows the graph of the proposed model. In the graph, nodes represent heterogeneous edge or end devices and the edge represents communication channel. The links between the nodes may be guided on unguided. In the context of cloud computing nodes may also represent the groups or clusters of the end devices [23], [24], [25]. The propose graph topology for distributed federated learning approach deploys the multiple edge (or end) devices like mobile phone, smart sensors etc. End device executes the process and the communication among the processes is performed through the message passing. Each edge device has its own model with local dataset [26], [27], [28], [29].

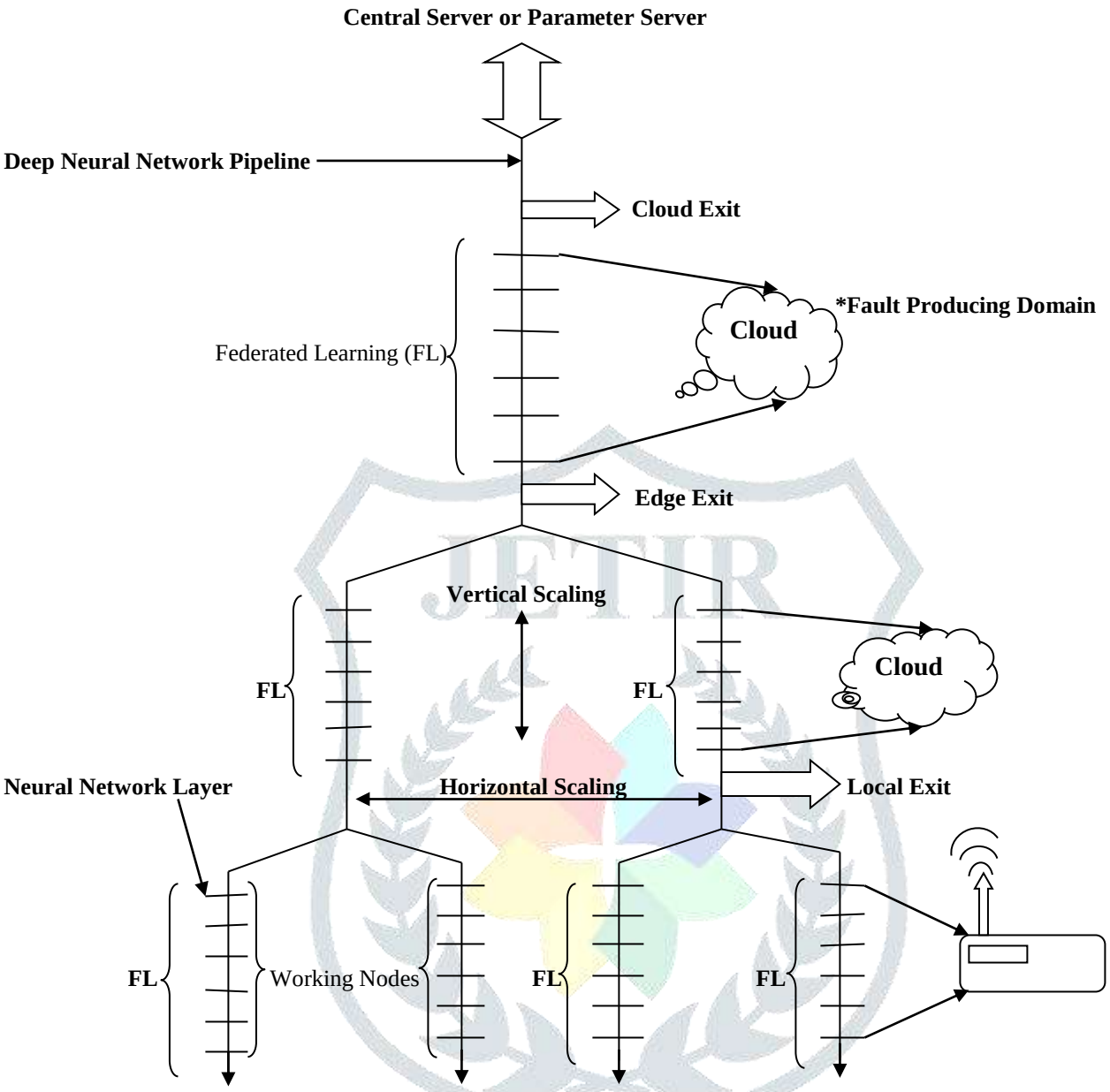

Figure 1: Architecture of Distributed Deep Neural Network with Federated Learning

Figure 1 shows the schematic diagram for distributed computing environment. Here each edge device may forms a cluster or computing cloud. Proposed methodology works on each edge device [30], [31], [32], [33] [34]. Edge devices are vast consumer or producer of big data In users' perspective the data at users' end is very confidential. Hence data security and privacy is paramount concern. In system perspective these edge or end devices may produce various kinds of faults in great extent [35], [36], [37], [38], [39], [40].

To make system reliable, fault resilient platform [41-46] must be implemented in the existing distributed system. Faults may be of two kinds: physical fault [47-56] and computational faults. The present work deal only computational faults [57-65].

3. Proposed Methodologies

FedLearning algorithm incorporates ensemble learning approach and establishes a coordination among multiple distributed working machines [66-74].

FedLearning algorithm

Algorithm: FedLearning

Input: Dataset x

Output: Fault Tolerant Model

1. Initialize model parameter set $\boldsymbol{\theta}$
2. **for** *i in iteration* **do**
3. **F**orward propagation: $\boldsymbol{output_i = fp\ (x_i\ ,\ \theta_i)}$;
4. Evaluate cost function: $\boldsymbol{c_i = cost\ (\alpha(x_i), output_i)}$;
5. **if** $c_i < \varepsilon$ **then**
6. Break;
7. **else**
8. Backword propagation: $\boldsymbol{gradient_i = bp\ (x_i\ ,\ \theta_i,\ c_i)}$;
9. Send edge devices' gradients to parameter computing central server and obtain new updated gradients;
10. Update: $\boldsymbol{\theta_{i+1} = \theta_i - \eta * gradient_{new}}$;
11. **end if**
12. **end for**
13. **return** Model with parameter set $\boldsymbol{\theta}$

Table 1. Description of parameters/hyperparameter used in FedLearning

S. No.	Parameter/Hyperparameter	Description
1	x_i	Data point at ith iteration
2	θ	Set of Model parameters
3	*fp*	Feedforward procedure
4	α	The activation function
5	*cost*	The cost or loss function
6	*output*	The output of each epoc
7	c_i	Cost evaluated by cost function on ith iteration
8	*bp*	Backward propagation procedure
9	ε	Error arbitrarily chosen very small
10	$gradient_i$	Gradient evaluated by *bp* process
11	$gradient_{new}$	Gradient evaluated by central computing server
12	η	Learning rate of training

The objective of this research work is to deal with the computational fault tolerance. The computational fault tolerance refers to a system which is tolerant to changes in the functionality occurring from the operation of the abstract components in the system being defective.

4. Experimental Setup

4.1 Benchmark Dataset

Appropriate datasets that can help to train, test and validate the fault prediction algorithm are rarely available or accessible for research work. It creates a major problem to the research process in the domain of fault tolerance and fault prevention [75-77]. Disk failure is a very common kind of failure in traditional computing system. This research uses backblaze hard drive failure rates dataset for training the proposed model.

Backblaze Hard Drive Failure Rates for 2020
(Reporting Period 01/01/2020 – 12/31/2020 inclusive)

MFG	Model	Drive Size	Drive Count	Avg Age (Months)	Drive Days	Drive Failures	AFR
HGST	**HMS5C4040ALE640**	**4TB**	**3,100**	**56.65**	**1,083,774**	**8**	**0.27%**
HGST	**HMS5C4040BLE640**	**4TB**	**12,744**	**50.43**	**4,663,049**	**34**	**0.27%**
HGST	**HMS5C4040ALE600**	**8TB**	**1,075**	**34.85**	**372,000**	**3**	**0.29%**
HGST	**HMS5C4040ALE600**	**12TB**	**2,600**	**15.04**	**820,272**	**7**	**0.31%**
HGST	**HMS5C4040ALE604**	**12TB**	**2,506**	**3.78**	**275,779**	**9**	**1.19%**
HGST	**HMS5C4040ALE604**	**12TB**	**10,830**	**21.01**	**3,968,475**	**50**	**0.46%**
Seagate	**ST4000DM000**	**4TB**	**18,939**	**62.35**	**6,983,470**	**269**	**1.41%**
Seagate	**ST6000X000**	**6TB**	**886**	**68.84**	**324,275**	**2**	**0.23%**
Seagate	**ST8000DM002**	**8TB**	**9,772**	**51.07**	**3,584,788**	**91**	**0.93%**
Seagate	**ST8000NM005**	**8TB**	**14,406**	**41.34**	**5,286,790**	**177**	**1.22%**
Seagate	**ST10000NM0086**	**10TB**	**1,201**	**38.73**	**439,247**	**16**	**1.33%**
Seagate	**ST12000NM0007**	**12TB**	**23,036**	**29.78**	**11,947,303**	**339**	**1.04%**
Seagate	**ST12000NM0008**	**12TB**	**19,287**	**9.76**	**5,329,149**	**148**	**1.01%**
Seagate	**ST12000NM001G**	**12TB**	**7,130**	**6.08**	**454,090**	**30**	**0.84%**
Seagate	**ST14000NM001G**	**14TB**	**5,987**	**2.89**	**5,784**	**13**	**1.04%**
Seagate	**ST14000NM0138**	**14TB**	**360**	**1.56**	**21,323**	**0**	**0.00%**
Seagate	**ST16000NM001G**	**16TB**	**59**	**12.93**	**5,820**	**1**	**1.71%**
Seagate	**ST18000NM000J**	**16TB**	**60**	**3.27**	**36,234**	**2**	**12.54%**
Toshiba	**MD04ABA400V**	**4TB**	**99**	**67.29**	**4,103,823**	**0**	**0.00%**
Toshiba	**MG07ACA14TA**	**14TB**	**21,046**	**7.65**	**2,562**	**102**	**0.91%**
Toshiba	**MG07ACA14TEY**	**14TB**	**160**	**1.22**	**33,774**	**0**	**0.00%**
Toshiba	**MG07ACA16TEY**	**16TB**	**1,014**	**2.14**	**33,774**	**0**	**0.00%**
WDC	**WUH721414ALE6L4**	**14TB**	**6,002**	**1.68**	**229,861**	**1**	**0.16%**
		TOTAL	**162,299**		**51,267,791**	**1,302**	**0.93%**

The proposed training algorithm also uses a benchmark data set which is retrieved from a Hadoop Distributed File System (HDFS).

4.2 Simulation Tools

The proposed algorithm is implemented by using Google Colab and CloudSim simulators. Python and R language have been used for model implementation.

5. Comparison with the Existing Model

The proposed model is compatible with tolerating the arbitrary faults occurring in distributed system. As compared to other model this algorithm produces very smaller training loss.

6. Conclusions and Discussion

In the proposed model the participating edge device trains its own model with local dataset. These local datasets are not sharable among the edge devices hence the system preserves the privacy- sensitive personal data. Federated learning collaborates with machine learning without centralized training of the data.

7. Future Work

Federated learning poses some of the key problems which have to be resolved: one of the problems is communication cost and other one is unreliability of the end devices that need not necessarily participate in the FL process. The proposed line of work opens the options for further research in direction of data security and privacy of the personal data.

8. References

[1] Y. LeCun, Y. Bengio, and G. Hinton, ‘‘Deep learning,’’ Nature, vol. 521, pp. 436–444, May 2015.

[2] B. Mohammed, I. Awan, H. Ugail, M. Younas, Failure prediction using machine learning in a virtualised HPC system and application, Cluster Computing. 22 pp. 471–485. https://doi.org/10.1007/s10586-019-02917-1, 2019.

[3] C. Alippi, ‘‘Selecting accurate, robust, and minimal feedforward neural networks,’’ IEEE Trans. Circuits Syst. I, Fundam. Theory Appl., vol. 49, no. 12, pp. 1799–1810, Dec. 2002.

[4] O. Beaumont, L. Eyraud-Dubois, J.-A. Lorenzo-del-Castillo, Analyzing real cluster data for formulating allocation algorithms in cloud platforms, Parallel Computing. 54 pp. 83–96. https://doi.org/10.1016/j.parco.2015.07.001, 2016.

[5] H. R. Mahdiani, S. M. Fakhraie, and C. Lucas, ‘‘Relaxed fault-tolerant hardware implementation of neural networks in the presence of multiple transient errors,’’ IEEE Trans. Neural Netw. Learn. Syst., vol. 23, no. 8, pp. 1215–1228, Aug. 2012.

[6] S. Srinivasan and C. F. Stevens, ‘‘Robustness and fault tolerance make brains harder to study,’’ BMC Biol., vol. 9, pp. 46, Jun. 2011.

[7] C.A. Rincon C., J.-F. Paris, R. Vilalta, A.M.K. Cheng, D.D.E. Long, Disk failure prediction in heterogeneous environments, in: 2017 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS), IEEE, Seattle, WA,: pp. 1–7. https://doi.org/10.23919/SPECTS.2017.8046776. 2017.

[8] W. Maass, ‘‘Noise as a resource for computation and learning in networks of spiking neurons,’’ Proc. IEEE, vol. 102, no. 5, pp. 860–880, May 2014.

[9] T. Sejnowksi and T. Delbruck, ‘‘The language of the brain,’’ Sci. Amer., vol. 307, pp. 54–59, Oct. 2012.

[10] K. Patan, Neural Network-Based Model Predictive Control: Fault Tolerance and Stability, IEEE Transactions on Control Systems Technology. 23 pp. 1147–1155. https://doi.org/10.1109/TCST.2014.2354981, 2015.

[11] W. Maass, ‘‘To spike or not to spike: That is the question,’’ Proc. IEEE, vol. 103, no. 12, pp. 2219–2224, Dec. 2015.

[12] P. Chandra and Y. Singh, ‘‘Fault tolerance of feedforward artificial neural networks—A framework of study,’’ in Proc. Int. Joint Conf. Neural Netw., vol. 1. Jul. 2003, pp. 489–494.

[13] P. Kumari, P. Kaur, A survey of fault tolerance in cloud computing, Journal of King Saud University - Computer and Information Sciences. https://doi.org/10.1016/j.jksuci.2018.09.021, 2018.

[14] J. A. G. Nijhuis and L. Spaanenburg, ‘‘Fault tolerance of neural associative memories,’’ IEE Proc. E-Comput. Digit. Techn., vol. 136, no. 5, pp. 389–394, Sep. 1989.

[15] S. S. Venkatesh, ‘‘The science of making ERORS: What error tolerance implies for capacity in neural networks,’’ IEEE Trans. Knowl. Data Eng., vol. 4, no. 2, pp. 135–144, Apr. 1992.

[16] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, ‘‘Neural acceleration for general-purpose approximate programs,’’ IEEE Micro, vol. 33, no. 3, pp. 16–27, May 2013.

[17] Z. Du, A. Lingamneni, Y. Chen, K. V. Palem, O. Temam, and C. Wu, ‘‘Leveraging the error resilience of neural networks for designing highly energy efficient accelerators,’’ IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 34, no. 8, pp. 1223–1235, Aug. 2015.

[18] G. Volanis, A. Antonopoulos, A. A. Hatzopoulos, and Y. Makris, ‘‘Toward silicon-based cognitive neuromorphic ICs—A survey,’’ IEEE Design Test, vol. 33, no. 3, pp. 91–102, Jun. 2016.

[19] J. Arlat, Z. Kalbarczyk, and T. Nanya, ‘‘Nanocomputing: Small devices, large dependability challenges,’’ IEEE Security Privacy, vol. 10, no. 1, pp. 69–72, Jan. 2012.

[20] Z. Wang, K. H. Lee, and N. Verma, ‘‘Overcoming computational errors in sensing platforms through embedded machine-learning kernels,’’ IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 23, no. 8, pp. 1459–1470, Aug. 2015.

[21] D. Terry, ‘‘Toward a new approach to IoT fault tolerance,’’ Computer, vol. 49, no. 8, pp. 80–83, Aug. 2016.

[22] J. von Neumann, ‘‘Probabilistic logics and the synthesis of reliable organisms from unreliable components,’’ Autom. Stud., vol. 34, pp. 43–98, 1956.

[23] I.P. Egwutuoha, S. Chen, D. Levy, B. Selic, A Fault Tolerance Framework for High Performance Computing in Cloud, in: IEEE, pp. 709–710. https://doi.org/10.1109/CCGrid.2012.80, 2012.

[24] D. D. Thaker, R. Amirtharajah, F. Impens, I. L. Chuang, and F. T. Chong, ‘‘Recursive TMR: Scaling fault tolerance in the nanoscale era,’’ IEEE Design Test Comput., vol. 22, no. 4, pp. 298–305, Jul. 2005.

[25] K. Roy, B. Jung, D. Peroulis, and A. Raghunathan, ‘‘Integrated systems in the more-than-Moore era: Designing low-cost energy-efficient systems using heterogeneous components,’’ IEEE Design Test, vol. 33, no. 3, pp. 56–65, Jun. 2016.

[26] M. Haselman and S. Hauck, ‘‘The future of integrated circuits: A survey of nanoelectronics,’’ Proc. IEEE, vol. 98, no. 1, pp. 11–38, Jan. 2010.

[27] A. Eghbal, P. M. Yaghini, N. Bagherzadeh, and M. Khayambashi, ‘‘Analytical fault tolerance assessment and metrics for TSV-based 3D network-on-chip,’’ IEEE Trans. Comput., vol. 64, no. 12, pp. 3591–3604, Dec. 2015.

[28] A. Rahimi, L. Benini, and R. K. Gupta, ‘‘Variability mitigation in nanometer CMOS integrated systems: A survey of techniques from circuits to software,’’ Proc. IEEE, vol. 104, no. 7, pp. 1410–1448, Jul. 2016.

[29] S. Mittal and J. S. Vetter, ‘‘A survey of techniques for modeling and improving reliability of computing systems,’’ IEEE Trans. Parallel Distrib. Syst., vol. 27, no. 4, pp. 1226–1238, Apr. 2016.

[30] A. DeHon, N. Carter, and H. Quinn. (Mar. 2011). ‘‘CCC cross-layer reliability visioning study,’’ Nat. Sci. Found., USA, Tech. Rep. LA-UR 10-08387. [Online]. Available: http://www. relxlayer.org

[31] P. M. Furth and A. G. Andreou, ‘‘On fault probabilities and yield models for VLSI neural networks,’’ IEEE J. Solid-State Circuits, vol. 32, no. 8, pp. 1284–1287, Aug. 1997.

[32] J. M. Shalf and R. Leland, ‘‘Computing beyond Moore’s law,’’ Computer, vol. 48, no. 12, pp. 14–23, Dec. 2015.

[33] A. E. Barbour and A. S. Wojcik, ‘‘A general constructive approach to fault-tolerant design using redundancy,’’ IEEE Trans. Comput., vol. 38, no. 1, pp. 15–29, Jan. 1989.

[34] M. Peercy and P. Banerjee, ‘‘Fault tolerant VLSI systems,’’ Proc. IEEE, vol. 81, no. 5, pp. 745–758, May 1993.

[35] V. Piuri, ‘‘Analysis of fault tolerance in artificial neural networks,’’ J. Parallel Distrib. Comput., vol. 61, no. 1, pp. 18–48, 2001.

[36] A.-D. Almasi, S. Wozniak, V. Cristea, Y. Leblebici, and T. Engbersen, ‘‘Review of advances in neural networks: Neural design technology stack,’’ Neurocomputing, vol. 174, pp. 31–41, Jan. 2016.

[37] N. C. Hammadi and H. Ito, ‘‘Improving the performance of feedforward neural networks by noise injection into hidden neurons,’’ J. Intell. Robot. Syst., vol. 21, no. 2, pp. 103–115, 1998.

[38] P. J. Edwards and A. F. Murray, “Toward optimally distributed computation,’’ Neural Comput., vol. 10, no. 4, pp. 987–1005, Sep. 1998.

[39] N. S. Merchawi, S. R. T. Kumara, and C. R. Das, ‘‘A probabilistic model for the fault tolerance of multilayer perceptrons,’’ IEEE Trans. Neural Netw., vol. 7, no. 1, pp. 201–205, Jan. 1996.

[40] X. Zeng and D. S. Yeung, ‘‘Sensitivity analysis of multilayer perceptron to input and weight perturbations,’’ IEEE Trans. Neural Netw., vol. 12, no. 6, pp. 1358–1366, Nov. 2001.

[41] M. Stevenson, R. Winter, and B. Widrow, ‘‘Sensitivity of feedforward neural networks to weight errors,’’ IEEE Trans. Neural Netw., vol. 1, no. 1, pp. 71–80, Mar. 1990.

[42] Y. L. Cun, J. S. Denker, and S. A. Solla, ‘‘Advances in neural information processing systems,’’ in Optimal Brain Damage, D. S. Touretzky, Ed. San Francisco, CA, USA: Morgan Kaufmann, 1990, pp. 598–605.

[43] J. L. Bernier, J. Ortega, E. Ros, I. Rojas, and A. Prieto, ‘‘A quantitative study of fault tolerance, noise immunity, and generalization ability of MLPs,’’ Neural Comput., vol. 12, no. 12, pp. 2941–2964, 2000.

[44] E. B. Tchernev, R. G. Mulvaney, and D. S. Phatak, ‘‘Investigating the fault tolerance of neural networks,’’ Neural Comput., vol. 17, no. 7, pp. 1646–1664, Jul. 2005.

[45] E. M. El Mhamdi and R. Guerraoui, ‘‘When neurons fail—Technical report,’’ EPFL, Lausanne, Tech. Rep. EPFL-WORKING-217561, 2016.

[46] Y. Wang and A. Avižienis, ‘‘A unified reliability model for fault-tolerant computers,’’ IEEE Trans. Comput., vol. C-29, no. 11, pp. 1002–1011, Nov. 1980.

[47] A. Kulakov, M. Zwolinski, and J. S. Reeve, ‘‘Fault tolerance in distributed neural computing,’’ CoRR, vol. abs/1509.09199, pp. 1–9, Sep. 2015.

[48] A. Avižienis, ‘‘Framework for a taxonomy of fault-tolerance attributes in computer systems,’’ SIGARCH Comput. Archit. News, vol. 11, no. 3, pp. 16–21, Jun. 1983.

[49] V. P. Nelson, ‘‘Fault-tolerant computing: Fundamental concepts,’’ Computer, vol. 23, no. 7, pp. 19–25, Jul. 1990.

[50] A. K. Somani and N. H. Vaidya, ‘‘Understanding fault tolerance and reliability,’’ Computer, vol. 30, no. 4, pp. 45–50, Apr. 1997.

[51] P. G. Depledge, ‘‘Fault-tolerant computer systems,’’ IEE Proc. A-Phys. Sci., Meas. Instrum., Manage. Edu.-Rev., vol. 128, no. 4, pp. 257–272, May 1981.

[52] G. Buja and R. Menis, ‘‘Dependability and functional safety: Applications in industrial electronics systems,’’ IEEE Ind. Electron. Mag., vol. 6, no. 3, pp. 4–12, Sep. 2012.

[53] S. Gai, M. Mezzalama, and P. Prinetto, ‘‘A review of fault models for LSI/VLSI devices,’’ Softw. Microsyst., vol. 2, no. 2, pp. 44–53, Apr. 1983.

[54] J. Sosnowski, ‘‘Transient fault tolerance in digital systems,’’ IEEE Micro, vol. 14, no. 1, pp. 24–35, Feb. 1994.

[55] P. Pop, V. Izosimov, P. Eles, and Z. Peng, ‘‘Design optimization of timeand cost-constrained fault-tolerant embedded systems with checkpointing and replication,’’ IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 17, no. 3, pp. 389–402, Mar. 2009.

[56] N. Aymerich, S. D. Cotofana, and A. Rubio, ‘‘Adaptive fault-tolerant architecture for unreliable technologies with heterogeneous variability,’’ IEEE Trans. Nanotechnol., vol. 11, no. 4, pp. 818–829, Jul. 2012.

[57] K. C. Y. Mei, ‘‘Bridging and stuck-at faults,’’ IEEE Trans. Comput., vol. C-23, no. 7, pp. 720–727, Jul. 1974.

[58] G. Bolt, ‘‘Fault models for artificial neural networks,’’ in Proc. IEEE Int. Joint Conf. Neural Netw., vol. 2. Nov. 1991, pp. 1373–1378.

[59] A. Pancholy, J. Rajski, and L. J. McNaughton, ‘‘Empirical failure analysis and validation of fault models in CMOS VLSI circuits,’’ IEEE Design Test Comput., vol. 9, no. 1, pp. 72–83, Mar. 1992.

[60] P. Gil, J. Arlat, H. Madeira, Y. Crouzet, T. Jarboui, and K. Kanoun, ‘‘Fault representativeness,’’ Eur. Community Dependability Benchmarking Project, France, Tech. Rep. IST-200025425, 2002.

[61] D. de Andres, J. C. Ruiz, D. Gil, and P. Gil, ‘‘Fault emulation for dependability evaluation of VLSI systems,’’ IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 16, no. 4, pp. 422–431, Apr. 2008.

[62] J. A. Abraham and W. K. Fuchs, ‘‘Fault and error models for VLSI,’’ Proc. IEEE, vol. 74, no. 5, pp. 639–654, May 1986.

[63] P. E. Dodd and L. W. Massengill, ‘‘Basic mechanisms and modeling of single-event upset in digital microelectronics,’’ IEEE Trans. Nucl. Sci., vol. 50, no. 3, pp. 583–602, Jun. 2003.

[64] S. Ghosh and K. Roy, ‘‘Parameter variation tolerance and error resiliency: New design paradigm for the nanoscale era,’’ Proc. IEEE, vol. 98, no. 10, pp. 1718–1751, Oct. 2010.

[65] M. Y. C. Kao, K.-T. Tsai, and S.-C. Chang, ‘‘A fault detection and tolerance architecture for post-silicon skew tuning,’’ IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 23, no. 7, pp. 1210–1220, Jul. 2015.

[66] M. Al-Kuwaiti, N. Kyriakopoulos, and S. Hussein, ‘‘A comparative analysis of network dependability, fault-tolerance, reliability, security, and survivability,’’ IEEE Commun. Surveys Tuts., vol. 11, no. 2, pp. 106–124, 2nd Quart., 2009.

[67] R. Frei, R. McWilliam, B. Derrick, A. Purvis, A. Tiwari, and G. Di Marzo Serugendo, ‘‘Self-healing and self-repairing technologies,’’ Int. J. Adv. Manuf. Technol., vol. 69, no. 5, pp. 1033–1061, 2013.

[68] M. A. Breuer, ‘‘Multi-media applications and imprecise computation,’’ in Proc. 8th Euromicro Conf. Digit. Syst. Des. (DSD), Aug. 2005, pp. 2–7.

[69] M. Stanisavljević, A. Schmid, and Y. Leblebici, Fault-Tolerant Architectures and Approaches. New York, NY, USA: Springer, 2011, pp. 35–47.

[70] E. Dubrova, Hardware Redundancy. New York, NY, USA: Springer, 2013, pp. 55–86.

[71] D. S. Phatak and I. Koren, ‘‘Complete and partial fault tolerance of feedforward neural nets,’’ IEEE Trans. Neural Netw., vol. 6, no. 2, pp. 446–456, Mar. 1995.

[72] C. Neti, M. H. Schneider, and E. D. Young, ‘‘Maximally fault tolerant neural networks,’’ IEEE Trans. Neural Netw., vol. 3, no. 1, pp. 14–23, Jan. 1992.

[73] P. W. Protzel, D. L. Palumbo, and M. K. Arras, ‘‘Performance and fault-tolerance of neural networks for optimization,’’ IEEE Trans. Neural Netw., vol. 4, no. 4, pp. 600–614, Jul. 1993.

[74] C. H. Sequin and R. D. Clay, ‘‘Fault tolerance in artificial neural networks,’’ in Proc. Int. Joint Conf. Neural Netw. (IJCNN), vol. 1. Jun. 1990, pp. 703–708.

[75] K. Mehrotra, C. K. Mohan, and S. Ranka, ‘‘Fault tolerance in neural networks,’’ School Comput. Inf. Sci., Syracuse Univ., Syracuse, NY, USA, Tech. Rep. RL-TR-94-93, Jul. 1994.

[76] Y. Tohma and Y. Koyanagi, ‘‘Fault-tolerant design of neural networks for solving optimization problems,’’ IEEE Trans. Comput., vol. 45, no. 12, pp. 1450–1455, Dec. 1996.

[77] D. B. I. Feltham and W. Maly, ‘‘Physically realistic fault models for analog CMOS neural networks,’’ IEEE J. Solid-State Circuits, vol. 26, no. 9, pp. 1223–1229, Sep. 1991.