Institut für Theoretische Elektrotechnik und Mikroelektronik

Universität Bremen

PhD proposal

Research project:
System-on-Chip Architectures for Real-Time Machine Learning Algorithms in Industrial Internet-of-Things Applications

Candidate name:
Yarib Israel Nevárez Esparza

March 22, 2021

1 General information

1.1 Personal information

First name: Yarib Israel
Last name: Nevárez Esparza
Date of birth: May 26, 1986
Phone: +49 (1) 788 936794
E-mail: nevarez@item.uni-bremen.de

Yarib Israel Nevárez Esparza

1.2 Supervisor

Title: Prof. Dr.-Ing.
First name: Alberto
Last name: García-Ortiz
Department: Fachbereich 1 - Physik / Elektrotechnik, Institut für theoretische Elektrotechnik und Mikroelektronik
Office address: Building NW 1, Otto-Hahn Allee 1, 28359 Bremen
Phone: +49 (0) 421 218 62533
E-mail: agarcia@item.uni-bremen.de

Prof. Dr.-Ing. Alberto García-Ortiz

1.3 Specialization and working direction

Specialization: Electrical and computer engineering
Working direction: Integrated digital systems

1.4 Expected total duration

The planned duration is 3 years. Start date: May 27, 2019. End date: May 27, 2022.

Abstract

In the emerging era of Industry 4.0, Machine Learning (ML) algorithms yield the power of Artificial Intelligence (AI) to the ubiquitous Internet of Things (IoT) devices. Applications in this field become smarter and more profitable as the availability of big data increases, driving the evolution of many aspects of daily life, science, and industry. However, state-of-the-art ML algorithms, specially Spiking Neural Networks (SNNs) and deep Convolutional Neural Networks (CNNs), are highly compute and data intensive. Therefore, the substantial demand for power and hardware resources of these algorithms represents a restriction for IoT devices in the scope of embedded systems.

Energy, performance, and resource utilization are the key design concerns in computer systems. Considering the intrinsic error resilience of ML algorithms, paradigms such as approximate computing come to the rescue by offering promising efficiency gains to assist the aforementioned design concerns. Approximation techniques are widely used in ML algorithms at the model-structure as well as at the hardware processing level. However, state-of-the-art approximate computing methodologies do not sufficiently address accelerator designs for Deep Neural Networks (DNN) as power- and resource-demanding algorithms.

To sustain the continuous expansion of ML applications on resource-constrained devices, approximate computing will gradually transform from a design alternative to an essential prerequisite. This PhD proposal focuses on the investigation of approximate computing techniques to exploit the intrinsic error resilience of ML algorithms to optimize computing embedded systems. The goal of this research is to contribute to state-of-the-art knowledge with formal methodologies to address hardware design for neural network accelerators based on approximate computing.

Furthermore, the expected outcome of this PhD is to develop high-efficiency neural network accelerator architectures with a common design methodology: (i) SNN accelerator, to assist research in Neurosciences; and (ii) deep CNN accelerator, to assist industrial computer vision applications (e.g., real-time multiple object detection). Finally, the motivation of this work is to support the growing demand of processing capabilities of ML algorithms in the scope of embedded systems, and to contribute to the rise of a sustainable power-efficient next generation of neural network accelerators based on approximate computing.

2 Introduction

Industry is the piece of an economy that produces material goods which are highly mechanized and automatized. Since the beginning of industrialization, technological leaps have led to paradigm shifts that are now called ”industrial revolutions”: from mechanization, electrification, and later, digitalization (the so-called 3rd industrial revolution). Based on the advanced digitalization within factories, the combination of Internet technologies and future-oriented technologies in the field of ”smart” things (machines and products) seems to result in a new fundamental paradigm shift in industrial production. Emerging from this future expectation, the term ”Industry 4.0” was established for an expected ”4th industrial revolution” [1].

2.1 Internet-of-Things in Industry

To build the emerging environment of Industry 4.0, disruptive technologies are required to handle autonomous communications between all industrial devices throughout the factory and the Internet. Such technologies offer the potential to transform the industry along the entire production chain and stimulate productivity and overall economic growth [2]. These technologies include cloud computing, big data, and specially a new generation of IoT devices fused with Cyber-Physical systems (CPS), augmented reality, ML analytics, and Artificial Intelligence (AI) in general [3].

2.2 Machine Learning Algorithms in Internet-of-Things

The continuous evolution of ML/AI algorithms and IoT devices has not only made various ML/AI applications the major workload running on these embedded devices, but ML/AI has become the main approach for industrial solutions, especially in the rise of Industry 4.0 [3]. In fact, there is a clear motivation to run ML/AI algorithms on IoT devices because of [4]: (1) feasibility of mission-critical real-time processing and inference; (2) privacy and security of data; (3) offline operation capability; and (4) stressed communication robustness. Hence, the traditional term of IoT has also been redefined as AI of Things (AIoT) to emphasize the impact of ML/AI on this technology [5].

2.3 Constraints

The problem lies in the fact that state-of-the-art ML/AI algorithms, particularly DNNs, are highly compute and data intensive. This represents significant computational challenges across the spectrum of computing hardware, specially in the scope of embedded systems [6]. One of the most deployed applications is computer vision using CNNs. Compared to the conventional image processing approaches, the CNN accuracy has improved significantly that by 2015, a human can no longer beat a computer in image classification [4]. The early development of CNNs before 2016 mainly focused on accuracy improvement without considering computational costs. While accuracy of deep CNN for image classification improved 24% between 2012 and 2016, the demand on hardware resources increased more than 10×. Starting from 2017, significant attention was paid to improve hardware efficiency in terms of compute power, memory bandwidth, and power consumption, while maintaining accuracy at a similar level to human perception [6]. Nonetheless, the state-of-the-art of CNN-based algorithms, such as multiple object detection models (e.g., SPP-net [7], SSD [8], Faster R-CNN [9], and

YOLOv4 [10]), are still unsuitable for the resource-limited nature of embedded systems [11, 12].

Consequently, the recent breakthroughs in ML applications have brought significant advancements in neural network processors [13]. These rapid evolution, however, came at the cost of an important demand for computational power. Hence, to bring the inference speed to an acceptable level, custom ASIC Neural Processing Units (NPUs) are becoming ubiquitous in both embedded and general purpose computing. NPUs perform several tera operations per second in a confined area. Therefore, they become subject to elevated on-chip power densities that rapidly result in excessive on-chip temperatures during operation [14]. These design efforts focused on power-hungry parallel computing techniques, yet unsustainable for resource-constrained devices. As a result, radical changes to conventional computing approaches are required in order to sustain and improve performance while satisfying mandatory energy and temperature constraints [15].

2.4 Alternatives

To overcome the problem, based on the error-resilience of ML algorithms, an evident solution is approximate computing. This computing paradigm has been used in a wide range of applications to increase the hardware computational efficiency[16]. For neural network applications, two main approximation strategies are used, namely network compression and classical approximate computing[17].

2.4.1 Network Compression and Quantization

Researchers focusing on embedded applications started lowering the precision of weights and activation maps to shrink the memory footprint of the large number of parameters representing DNNs, a method known as network quantization. In this manner, reduced bit precision causes a small accuracy loss [18, 19, 20, 21]. In addition to quantization, network pruning reduces the model size by removing structural portions of the parameters and its associated computations [22, 23]. This method has been identified as an effective technique to improve the efficiency of CNN for applications with limited computational budget[24, 25, 26]. These techniques leverage the intrinsic error-tolerance of neural networks, as well as their ability to recover from accuracy degradation while training.

2.4.2 Approximate Computing

Approximate computing is an emerging design paradigm that is able to tradeoff computation quality (e.g., accuracy) and computational efficiency (e.g., in run-time, chip-area, and/or energy) by exploiting the error-resilience properties of applications [15, 27]. Data redundancy of neural networks incorporate a certain degree of resilience against random external and internal perturbations, for instance, random hardware faults.

This property can be exploited in a cross-layer resilience approach [28]: by leveraging error-resilience at algorithmic-level, it can be allowed a certain degree of inaccuracies at the computing-level. This approach consists of designing processing elements that approximate their computation by employing cleverly modified algorithmic logic units [16].

Approximate computing techniques allow substantial enhancement in processing efficiency with moderated accuracy degradation. Some research papers have shown the feasibility of applying approximate computing to the inference stage of neural networks [29, 16, 30, 31, 32, 33]. Such techniques usually demonstrated small inference accuracy degradation, but significant enhancement in computational performance, resource utilization, and energy consumption. Hence, by taking advantage of the intrinsic error-tolerance of neural networks, approximate computing is positioned as a promising approach for inference on resource-limited devices. Nonetheless, the complex state-of-the-art of CNN-based algorithms has not been sufficiently addressed with approximate computing techniques.

2.5 Problem Statement

While the state-of-the-art approximate computing techniques have presented highly-efficient adders and multipliers, they do not sufficiently address accelerator designs for ML algorithms, specifically neural networks.

2.6 Research Objective

Considering the broad range of ML algorithms, te research objective for this PhD proposal is the following: *Investigating formal design methodologies for high-efficiency neural network hardware accelerator based on digital approximate computing in the scope of embedded systems.*

2.6.1 Research Questions

- ◇ How to analyze neural networks for error resilience?
- ◇ How to exploit intrinsic error resilience of neural networks effectively?
- ◇ How to design neural network accelerators based on approximate computing?
- ◇ Considering the case study of SNNs applied in fundamental research, how do the proposed approximate computing methodology affects the quality and efficiency of the processing?
- ◇ Considering the case study of CNNs applied in industrial computer vision, how do the proposed approximate computing methodology affects the quality and efficiency of the processing?

- ◇ What are the possibilities and challenges to embrace approximate computing for neural network accelerators?
- ◇ What are the possibilities and challenges to embrace approximate computing for neural network accelerators in safety-critical applications?

2.7 Motivations

2.7.1 Fundamental Research

1. Derive formal methodologies to address hardware design for resource- and energy-efficient neural network accelerators based on approximate computing.
2. Contribute to de foundations of approximate neural network computing.

2.7.2 Expected Benefits

1. A sustainable development/expansion of resource-hungry ML algorithms in embedded systems.
2. An energy-efficient industrial environment.

3 State-of-the-art

3.1 Neural Network Resilience Analysis

3.1.1 Experimental Analysis

The majority of the research on error resilience in neural networks have been experimental. They range from physical fault induction experiments in real hardware devices [34, 35], over fault injections in (virtual) hardware models [36, 37, 38, 34], to error simulations at the algorithmic behavior level [39, 40, 41]. Behavioral analysis can be connected to realistic hardware faults in a second step, by mapping the effect of these faults to error models in the algorithm domain [42].

Experimental research found different determinants of neural network resilience, the most important being the number and type of errors, the data representation of the neural network, the DNN type, and the location where the error occurs. However, while experimental evaluation is useful for an accurate a posteriori resilience determination of a given DNN on hardware, it is cumbersome and provides only limited insight into a priori design choices for DNN developers to improve resilience at the algorithm level [43].

3.1.2 Theoretical Analysis

A theory-guided resilience analysis offers the advantage of being more directly interpretable and avoids lengthy fault injection experiments. El Mhamdi and Guerraoui [44] analytically derived easily computable bounds for the forward error propagation of neurons that are stuck-at-zero (crashed neurons) and for neurons that transmit arbitrary values (Byzantine neurons). They found that the choice of activation function and the number of neurons per layer are design choices that affect the forward error propagation. More precisely, an activation function with a low Lipschitz constant as well as a high number of neurons per layer can reduce forward error propagation.

A different analytical technique to derive neuron resilience prediction has been used in the context of approximate neural network computing. Backpropagation of error gradients, comparable to the technique used to determine weight updates during neural network training, has been used to estimate the average output sensitivity to perturbations in individual neurons [45, 27].

Recently, Schorn et al. [41] showed that a technique based on layerwise relevance propagation (LRP) [46] outperforms gradient-based resilience prediction. Contrarily to gradient methods, which determine the sensitivity to small perturbations in neurons, LRP attributes to each neuron its absolute contribution to the DNN output [47], which can be interpreted as layerwise Taylor decomposition [48]. A high neuron relevance, averaged over a training set of input samples, corresponds to a high sensitivity against errors [41].

3.2 Approximate Computing in Adders and Multipliers

Driven by this high potential for power reduction, designing approximate circuits has attracted significant research interest. At the custom hardware level, approximate computing targets mainly arithmetic units [49, 50, 51, 52] (e.g., adders and multipliers) since they form the core components of all computations and a vast number of error-tolerant applications. Specifically, in neural network inference the majority of the energy is consumed in the multiplication operations. Recent research showed that employing approximate multipliers in neural network inference can deliver significant energy savings for a minimal loss in accuracy [52, 53, 54]. However, designing approximate circuits under quality constraints heavily increases the design time cycle since the designer has to verify both functionality and optimality as well as operating within error bounds [55]. This task becomes even more challenging as the circuits complexity increases. To this end, several research activities, such as approximate high-level synthesis (AHLS) [56], focus on automating the generation of approximate circuits. Approximate HLS estimates error propagation and distributes the available error budget to the different approximate sub-components of a larger accelerator, such as convolution operators and generic matrix multiply units. As a result, AHLS enables generating complex approximate micro-architectures that satisfy given quality requirements.

Moreover, approximate computing is further subdivided into static and dynamically reconfigurable approximation techniques [33]. The latter, leveraging that error-tolerance and the induced errors are context- and input-dependent, aim to improve accuracy by providing a fine grain quality control and/or to further boost (power, energy, and/or run-time) gains by applying more aggressive approximation on less-sensitive inputs. Finally, reconfigurable approximation was also recently applied to address thermal constraints [14]. Instead of addressing thermal emergencies by reducing performance, by reducing the accuracy and hence dynamic power in the same area, the circuits power density decreases, resulting in lower temperatures.

3.3 Approximate Artificial Neural Network

With many approximate arithmetic building blocks presented in the literature (e.g., [49, 50, 51, 52]), some recent works proposed to design approximate neural networks with approximate neuron designs. In [30], Du et al. first designed approximate neurons with approximate multipliers, and then search a subset of substitutions to choose the best configuration for the approximate neural networks design. However, this is a trial-and-error process instead of a systematic solution.

Recently, Venkataramani et al. [45] proposed a systematic approximate neural network design methodology, namely AxNN. In this work, the final error of the NN is first back propagated to get error apportions of each individual neuron. Neurons are then sorted based on the magnitude of their average error contribution, and classified as resilient or sensitive ones with a pre-determined threshold. Those resilient neurons are then designed as approximate ones for energy savings. Finally, retraining is performed to mitigate the impact of inexact hardware on the solution quality.

4 Research goals

The goal of this research is to contribute to state-of-the-art knowledge with formal methodologies to address hardware design for energy-efficient neural network accelerators based on approximate computing.

4.1 Outcomes

The expected outcome of this PhD is to derive formal methodologies for neural network accelerators based on approximate computing. As a demonstration, it is expected to develop two neural network accelerators based on approximate computing:

- ◇ SNN accelerator to assist research in Neurosciences.
- ◇ Deep CNN accelerator to assist industrial computer vision applications (e.g., real-time multiple object detection).

5 Project plan

This PhD is divided into three phases that are subdivided into tasks. The prospective milestone schedule is shown in **Tab.** 1. The total timeframe is expected to be 3 years.

5.1 Phase 1 (completed)

1. Research of hardware design methodologies for custom neural network accelerators in embedded FPGA.
2. Design and implementation of an scalable neural network accelerator framework for design exploration.
3. Implement an spiking neural network model using the accelerator framework.
4. Report experimental results in a conference publication.

5.2 Phase 2 (completed)

1. Investigate error resilience of the SNN (Spike-by-Spike neural network).
2. Derive approximate computing methodology to exploit the error resilience of the SNN.
3. Implement the approximate computing approach on the neural network accelerator framework on FPGA.
4. Report approximate computing approach, methodology, and experimental results in a journal publication.

5.3 Phase 3

1. Extend the design exploration framework to support deep CNN.
2. Extend the design exploration framework to support CNN-based multiple object detection for computer vision.
3. Implement an state-of-the-art multiple object detection algorithm (e.g., You only look once (YOLO)) on FPGA.
4. Investigate error resilience analysis of CNN-based detection algorithm.
5. Implement hardware accelerator with the proposed approximate computing approach and design methodology.
6. Report approximate computing approach, methodology, and experimental results in a journal publication.

5.4 Phase 4

1. PhD thesis writing.

Milestone	Date	Description
M1	September, 2019	Completion of literature search
M2	January, 2020	Understand the key design considerations for efficient ML processing; understand trade-offs between various hardware architectures and platforms; learn about micro-architectural knobs such as precision, data reuse, and parallelism to architect ML accelerators given target area-power-performance metrics; evaluate the utility of various ML dataflow techniques for efficient processing; and understand future trends and opportunities of approximate neural network accelerators
M3	April, 2020	Outcome: Development of an accelerator framework of Spike-By-Spike (SbS) neural networks for inference and incremental learning in FPGA. Report results in a conference publication [57]
M4	July, 2020	Investigate error resilience analysis of SbS neural network algorithm
M5	October, 2020	Investigate and simulate approximate computing methodologies on SbS neural networks as a case study
M6	January, 2021	Report methodology and results of the case study in a journal publication
M7	April, 2021	Outcome: Extend the neural network accelerator framework to support deep CNN, and CNN-based multiple object detector algorithms for computer vision
M8	July, 2021	Investigate error resilience in CNN-based algorithms
M9	October, 2021	Outcome: Report results of methodology in a journal publication
M10	Jun, 2022	Written elaboration is completed

Table 1: Milestone schedule.

References

[1] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, “Industry 4.0,” *Business & information systems engineering*, vol. 6, no. 4, pp. 239–242, 2014.

[2] H. Espinoza, G. Kling, F. McGroarty, M. O’Mahony, and X. Ziouvelou, “Estimating the impact of the internet of things on productivity in europe,” *Heliyon*, vol. 6, no. 5, p. e03935, 2020.

[3] V. Alcácer and V. Cruz-Machado, “Scanning the industry 4.0: A literature review on technologies for manufacturing systems,” *Engineering science and technology, an international journal*, vol. 22, no. 3, pp. 899–919, 2019.

[4] K.-H. L. Loh, “1.2 fertilizing aiot from roots to leaves,” in *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 2020, pp. 15–21.

[5] J. Zhang and D. Tao, “Empowering things with intelligence: A survey of the progress, challenges, and opportunities in artificial intelligence of things,” *IEEE Internet of Things Journal*, 2020.

[6] S. Venkataramani, K. Roy, and A. Raghunathan, “Efficient embedded learning for iot devices,” in *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2016, pp. 308–311.

[7] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial pyramid pooling in deep convolutional networks for visual recognition,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[8] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[9] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: towards real-time object detection with region proposal networks,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 6, pp. 1137–1149, 2016.

[10] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “Yolov4: Optimal speed and accuracy of object detection,” *arXiv preprint arXiv:2004.10934*, 2020.

[11] I. Ahmad, S. Shahabuddin, T. Kumar, E. Harjula, M. Meisel, M. Juntti, T. Sauter, and M. Ylianttila, “Challenges of ai in wireless networks for iot,” *arXiv preprint arXiv:2007.04705*, 2020.

[12] F. Al-Turjman, *Artificial intelligence in IoT*. Springer, 2019.

[13] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers *et al.*, “In-datacenter performance analysis of a tensor processing unit,” in *Proceedings of the 44th annual international symposium on computer architecture*, 2017, pp. 1–12.

[14] H. Amrouch, G. Zervakis, S. Salamin, H. Kattan, I. Anagnostopoulos, and J. Henkel, “Npu thermal management,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 3842–3855, 2020.

[15] S. G. A. Gillani, “Exploiting error resilience for hardware efficiency: targeting iterative and accumulation based algorithms,” 2020.

[16] J. Han and M. Orshansky, “Approximate computing: An emerging paradigm for energy-efficient design,” in *2013 18th IEEE European Test Symposium (ETS)*. IEEE, 2013, pp. 1–6.

[17] M. Bouvier, A. Valentian, T. Mesquida, F. Rummens, M. Reyboz, E. Vianello, and E. Beigne, “Spiking neural networks hardware implementations and challenges: A survey,” *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 15, no. 2, pp. 1–35, 2019.

[18] M. Courbariaux, Y. Bengio, and J.-P. David, “Binaryconnect: Training deep neural networks with binary weights during propagations,” in *Advances in neural information processing systems*, 2015, pp. 3123–3131.

[19] S. Han, H. Mao, and W. J. Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding,” *arXiv preprint arXiv:1510.00149*, 2015.

[20] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, “Quantized neural networks: Training neural networks with low precision weights and activations,” *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6869–6898, 2017.

[21] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, “Xnor-net: Imagenet classification using binary convolutional neural networks,” in *European conference on computer vision*. Springer, 2016, pp. 525–542.

[22] Y. LeCun, J. Denker, and S. Solla, “Optimal brain damage,” *Advances in neural information processing systems*, vol. 2, pp. 598–605, 1989.

[23] B. Hassibi and D. Stork, “Second order derivatives for network pruning: Optimal brain surgeon,” *Advances in neural information processing systems*, vol. 5, pp. 164–171, 1992.

[24] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, “Pruning convolutional neural networks for resource efficient inference,” *arXiv preprint arXiv:1611.06440*, 2016.

[25] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, “Pruning filters for efficient convnets,” *arXiv preprint arXiv:1608.08710*, 2016.

[26] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, “Rethinking the value of network pruning,” *arXiv preprint arXiv:1810.05270*, 2018.

[27] Q. Zhang, T. Wang, Y. Tian, F. Yuan, and Q. Xu, “Approxann: An approximate computing framework for artificial neural network,” in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2015, pp. 701–706.

[28] N. P. Carter, H. Naeimi, and D. S. Gardner, “Design techniques for cross-layer resilience,” in *2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010)*. IEEE, 2010, pp. 1023–1028.

[29] U. Lotrič and P. Bulić, “Applicability of approximate multipliers in hardware neural networks,” *Neurocomputing*, vol. 96, pp. 57–65, 2012.

[30] Z. Du, K. Palem, A. Lingamneni, O. Temam, Y. Chen, and C. Wu, “Leveraging the error resilience of machine-learning applications for designing highly energy efficient accelerators,” in *2014 19th Asia and South Pacific design automation conference (ASP-DAC)*. IEEE, 2014, pp. 201–206.

[31] V. Mrazek, S. S. Sarwar, L. Sekanina, Z. Vasicek, and K. Roy, “Design of power-efficient approximate multipliers for approximate artificial neural networks,” in *Proceedings of the 35th International Conference on Computer-Aided Design*, 2016, pp. 1–7.

[32] S. S. Sarwar, S. Venkataramani, A. Raghunathan, and K. Roy, “Multiplier-less artificial neurons exploiting error resiliency for energy-efficient neural computing,” in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2016, pp. 145–150.

[33] G. Zervakis, H. Saadat, H. Amrouch, A. Gerstlauer, S. Parameswaran, and J. Henkel, “Approximate computing for ml: State-of-the-art, challenges and visions,” in *Proceedings of the 26th Asia and South Pacific Design Automation Conference*, 2021, pp. 189–196.

[34] F. F. dos Santos, P. F. Pimenta, C. Lunardi, L. Draghetti, L. Carro, D. Kaeli, and P. Rech, “Analyzing and increasing the reliability of convolutional neural networks on gpus,” *IEEE Transactions on Reliability*, vol. 68, no. 2, pp. 663–677, 2018.

[35] P. N. Whatmough, S. K. Lee, D. Brooks, and G.-Y. Wei, "Dnn engine: A 28-nm timing-error tolerant sparse deep neural network processor for iot applications," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 9, pp. 2722–2731, 2018.

[36] A. Azizimazreah, Y. Gu, X. Gu, and L. Chen, "Tolerating soft errors in deep learning accelerators with reliable on-chip memory designs," in *2018 IEEE International Conference on Networking, Architecture and Storage (NAS)*. IEEE, 2018, pp. 1–10.

[37] G. Li, S. K. S. Hari, M. Sullivan, T. Tsai, K. Pattabiraman, J. Emer, and S. W. Keckler, "Understanding error propagation in deep learning neural network (dnn) accelerators and applications," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2017, pp. 1–12.

[38] B. Salami, O. S. Unsal, and A. C. Kestelman, "On the resilience of rtl nn accelerators: Fault characterization and mitigation," in *2018 30th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*. IEEE, 2018, pp. 322–329.

[39] J. Marques, J. Andrade, and G. Falcao, "Unreliable memory operation on a convolutional neural network processor," in *2017 IEEE International Workshop on Signal Processing Systems (SiPS)*. IEEE, 2017, pp. 1–6.

[40] B. Reagen, U. Gupta, L. Pentecost, P. Whatmough, S. K. Lee, N. Mulholland, D. Brooks, and G.-Y. Wei, "Ares: A framework for quantifying the resilience of deep neural networks," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. IEEE, 2018, pp. 1–6.

[41] J. Li, G. Yan, W. Lu, S. Jiang, S. Gong, J. Wu, and X. Li, "Smartshuttle: Optimizing off-chip memory accesses for deep learning accelerators," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2018, pp. 343–348.

[42] V. Piuri, "Analysis of fault tolerance in artificial neural networks," *Journal of Parallel and Distributed Computing*, vol. 61, no. 1, pp. 18–48, 2001.

[43] C. Schorn, T. Elsken, S. Vogel, A. Runge, A. Guntoro, and G. Ascheid, "Automated design of error-resilient and hardware-efficient deep neural networks," *Neural Computing and Applications*, vol. 32, no. 24, pp. 18 327–18 345, 2020.

[44] R. Guerraoui *et al.*, "When neurons fail," in *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2017, pp. 1028–1037.

[45] S. Venkataramani, A. Ranjan, K. Roy, and A. Raghunathan, "Axnn: Energy-efficient neuromorphic systems using approximate computing," in *2014*

IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED). IEEE, 2014, pp. 27–32.

[46] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PloS one*, vol. 10, no. 7, p. e0130140, 2015.

[47] G. Montavon, W. Samek, and K.-R. Müller, "Methods for interpreting and understanding deep neural networks," *Digital Signal Processing*, vol. 73, pp. 1–15, 2018.

[48] G. Montavon, S. Lapuschkin, A. Binder, W. Samek, and K.-R. Müller, "Explaining nonlinear classification decisions with deep taylor decomposition," *Pattern Recognition*, vol. 65, pp. 211–222, 2017.

[49] J. Miao, K. He, A. Gerstlauer, and M. Orshansky, "Modeling and synthesis of quality-energy optimal approximate adders," in *Proceedings of the International Conference on Computer-Aided Design*, 2012, pp. 728–735.

[50] M. Shafique, W. Ahmad, R. Hafiz, and J. Henkel, "A low latency generic accuracy configurable adder," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2015, pp. 1–6.

[51] G. Zervakis, K. Koliogeorgi, D. Anagnostos, N. Zompakis, and K. Siozios, "Vader: Voltage-driven netlist pruning for cross-layer approximate arithmetic circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 6, pp. 1460–1464, 2019.

[52] H. Saadat, H. Bokhari, and S. Parameswaran, "Minimally biased multipliers for approximate integer and floating-point multiplication," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2623–2635, 2018.

[53] S. S. Sarwar, S. Venkataramani, A. Ankit, A. Raghunathan, and K. Roy, "Energy-efficient neural computing with approximate multipliers," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 14, no. 2, pp. 1–23, 2018.

[54] Z.-G. Tasoulas, G. Zervakis, I. Anagnostopoulos, H. Amrouch, and J. Henkel, "Weight-oriented approximation for energy-efficient neural network inference accelerators," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 12, pp. 4670–4683, 2020.

[55] G. Zervakis, S. Xydis, D. Soudris, and K. Pekmestzi, "Multi-level approximate accelerator synthesis under voltage island constraints," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 66, no. 4, pp. 607–611, 2018.

[56] S. Lee, L. K. John, and A. Gerstlauer, “High-level synthesis of approximate hardware under joint precision and voltage scaling,” in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE, 2017, pp. 187–192.

[57] Y. Nevarez, A. Garcia-Ortiz, D. Rotermund, and K. R. Pawelzik, “Accelerator framework of spike-by-spike neural networks for inference and incremental learning in embedded systems,” in *2020 9th International Conference on Modern Circuits and Systems Technologies (MOCAST)*. IEEE, 2020, pp. 1–5.