Low-Voltage Energy Efficient Neural Inference by Leveraging Fault Detection Techniques

Mehdi Safarpour[1], Tommy Z. Deng[2], John Massingham[2], Lei Xun[3], Mohammad Sabokrou[1&4], and Olli Silvén[1]
[1]Center for Machine Vision and Signal Analysis, University of Oulu, Finland
[2]Huawei Technologies Sweden AB, Stockholm, Sweden
[3]University of Southampton, Southampton, UK
[4]Institute for Research in Fundamental Sciences (IPM), Tehran, Iran
mehdi.safarpour@oulu.fi

***Abstract*—Operating at reduced voltages offers substantial energy efficiency improvement but at the expense of increasing the probability of computational errors due to hardware faults. In this context, we targeted Deep Neural Networks (DNN) as energy hungry building blocks in embedded systems. Without an error feedback mechanism, blind voltage down-scaling will result in degraded accuracy or total system failure. In this paper, solutions based on the inherent properties of Self-Supervised Learning (SSL) and Algorithm Based Fault Tolerance (ABFT) techniques were investigated. A DNN model trained on MNIST data-set was deployed on a Field Programmable Gate Array (FPGA) that operated at reduced voltages and employed the proposed schemes. The SSL approach provides extremely low-overhead fault detection at the cost of lower error coverage and extra training, while ABFT incurs less than 8% overheads at run-time with close to 100% error detection rate. By using the solutions, substantial energy savings, i.e., up to 48%, without compromising the accuracy of the model was achieved.**

***Index Terms*—Low power design, Self-supervised learning, Fault detection, FPGA, Neural Network, Radio resource management, 5G.**

I. Introduction

Arrival of 5G and 6G radio access technologies that aim to support millimeter wave bands up to 300 GHz and multi-gigabit per second data rates, necessitates deployment of high-performance and energy efficient wireless infrastructure to overcome the stringent thermal constraints. On the other hand, Deep Neural Networks (DNN) have proved to efficiently tackle the challenging Radio Resource Management (RRM) tasks in 5G base-stations [1], [2]. Besides, neural networks are emerging as essential building blocks of many applications in consumer electronic devices, demanding substantial energy and computational resources. Training and optimizing even a medium size neural model is computationally expensive [3], [4]. While neural inference consumes less energy, it is an equally important optimization target due to the large number of devices [5].

Tailoring hardware processors specifically for neural computations [1] is an approach for optimizing the energy efficiency. For instance, in [6] an Application Specific Processor (ASP) was proposed for neural inference by minimizing the instruction decoding effort and instead dedicating resources to actual computations. Other works, e.g., [7], have focused on algorithm level modifications of the models using quantization and pruning [4] to reduce the computational resource demands in return for a minor drop in model performance [8].

Since, the energy dissipation of digital logic is quadratically proportional to the supply voltage, in this work we are seeking solutions that enable reduced voltage operation without trading-off the performance of the model. Nonetheless, it is challenging to ensure the correctness of computations when the supply voltage is reduced. Timing uncertainties exacerbate at reduced voltages due to increased impact of process variations. Solutions based on static calibration or emulation are either too conservative, i.e., lead to unnecessary clock scaling, or do not take into account dynamic variations such as voltage noise, aging and temperature [9].

In GreenTPU [10] a neural inference accelerator with aggressive voltage scaling down to near-threshold voltage of transistors was designed and studied. Timing violations were shown to occur in Multiplier and Accumulator (MAC) units. Consequently, the MAC units were equipped with Timing Error Detection (TED) circuits. However, augmenting the design with TEDs increases substantially the design time and run-time overheads [11].

This calls for a mechanisms to detect errors on-the-fly without requiring modifications to the underlying hardware. Conventional system level fault detection approaches such as N-modular redundancy (NMR) increase the design size and power consumption by at least by a factor of two, rendering them unsuitable to be used for low-power purposes [12], [13]. In [14] the similarity of two consecutive video frames is exploited, expecting the prediction from the neural network to be the same for both. This simple fault-detection method doesn't incur overheads, however, the use cases for the technique is limited to video processing applications [15].

This paper presents two solutions for the detection of computational errors in neural inference deployed on an embedded platforms. Self-supervised Learning (SSL) [16] and Algorithm Based Fault Tolerance (ABFT) [17] were utilized as low-cost error detection schemes. While ABFT provides a high level of confidence in error detection as shown in [18] and [17], the SSL based scheme does not require modifications of the

* Source codes used to generate models in this study are available from https://github.com/NeuroFan/LowVoltageDNNonFPGA.git

implementation.

II. Proposed Solutions

We propose to add low-overhead redundancies to the neural model, either through SSL or ABFT, or both, to enable reliable dynamic voltage scaling (DVS). The computational errors are detected on-the-fly while the voltage is adjusted according to the detections, as shown in Fig. 1.

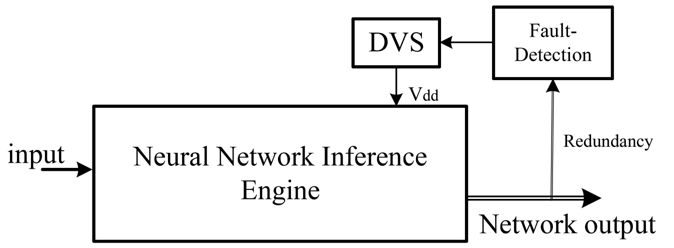


Figure 1. System level error detection enable aggressive voltage down-scaling.

As the first approach, we investigate a SSL based technique to detect accuracy degradation of the model due to reduced voltage induced computational errors. The SSL method was exploited by assigning a pre-text task to the model during the training. That task is made independent of the label of input data, however, it relies on computations common with the main task. Since the prediction accuracy of the dummy task is known at run-time, faulty operation of the network is detected at reduced voltage by monitoring the performance for the dummy task [19]. Neural networks exhibit some degree of error resilience, therefore, not all faults may show up.

The second solution based on ABFT detects computational errors at a finer-grained level in matrix arithmetic. While providing for more sensitive error detection [18], minor hardware modifications are necessary to make this approach applicable with a neural network.

A. Self-Supervised Learning

The idea behind the SSL is to automatically generate a supervisory signal to learn a discriminative and rich representation from the unlabeled data. The learned representation can be exploited effectively for solving the downstream tasks. Recent investigations have shown that SSL can improve the generalization and robustness of supervised task [19]. This means, learning a DNN for a supervised task (i.e., main task) such as classification will be more robust if it tries to optimize a SSL task (pre-text task) [20], [16] at the same time.

The DNN tries to solve the pre-text task by optimizing the generated (i.e., defined) supervision signal. In this paper, in the manner devised in [20], [16] the pre-text task is used to predict the rotation of the input image (See Fig.2) while the main task is considered to be image classification.

In the starting phase, the fed samples are rotated by pre-determined amounts, e.g., by 45 degrees, and the network, in addition to the main task, is asked to predict the amount of rotation. In this setting only the performance/accuracy of the SSL task is available at run-time, while there are no labeled samples for the main task in inference phase. Based on experimentation there is a strong correlation between the performances of both tasks. Consequently, we use the pre-text task for detecting the fault of the model for the main task.

A persisting failure of the network in the prediction of the rotation can indicate faulty operation of the model. Using this approach, no modification on generated network model is required.

It is important to notice that the overheads and fault detection capability of this approach depends on the design of the network and the pre-text task. Our aim was at demonstrating the utility of the scheme. A simple pre-text task as the one used in the current experiments incurs negligible overheads in the inference phase.

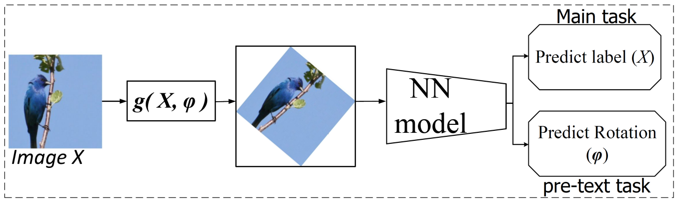


Figure 2. Process of self-supervised learning and detection. $g(.)$, X and φ are transform function, the input sample and amount of rotation, respectively.

B. Algorithm Based Fault Tolerance

ABFT proposed by Huang and Abraham [17] is a low-overhead method that enables detection of computational errors in large-scale supercomputing systems. ABFT methods cover all basic linear algebra operations. The majority of applications including DNNs rely upon matrix arithmetic [21].

When ABFT is used with matrix multiplication [22], the operand matrices are augmented with *checksum property* that is used to pinpoint computational errors in the output. Assuming A is an $N-$squared matrix, a *row checksum* matrix A^r is defined as a $N \times (N+1)$ matrix, as following [22].

$$A^r = \begin{bmatrix} A & Ae^T \end{bmatrix} \tag{1}$$

where $e_N = [1, 1, ..., 1]$. The *column checksum*, A^c, and *full checksum*, A^f matrices are defined in a similar manner, see Equations 2 and 3, respectively.

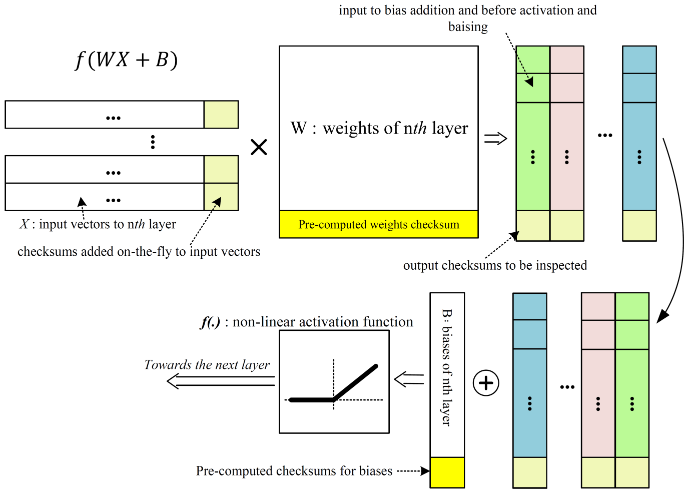
$$A^c = \begin{bmatrix} A \\ eA \end{bmatrix} \tag{2}$$

$$A^f = \begin{bmatrix} A & Ae^T \\ eA & eAe^T \end{bmatrix} \tag{3}$$

In case of matrix multiplication, ABFT ensures the existence of checksum property in result matrix when the column checksum matrix, A^r, is multiplied in a row checksum matrix as Eq. 4. Inconsistencies in checksums reveal the existence of a computational error.

$$C^f = \begin{bmatrix} A \\ eA \end{bmatrix} \begin{bmatrix} B & Be^T \end{bmatrix} = \begin{bmatrix} AB & ABe^T \\ eAB & eABe^T \end{bmatrix} \tag{4}$$

The overhead ratio of ABFT method grows sub-linearly with matrix size. This is an advantage when used within the layers with large number of inputs and outputs [18], [23].



$f(WX + B)$
input to bias addition and before activation and baising
W : weights of nth layer
Pre-computed weights checksum
X : input vectors to nth layer
checksums added on-the-fly to input vectors
output checksums to be inspected
f(.) : non-linear activation function
Towards the next layer
B: biases of nth layer
Pre-computed checksums for biases

Figure 3. Checksum of inputs to each layer is added on-the-fly, while checksums of weights are pre-computed. Multiplication and bias implementations are carried out using ABFT based arithmetic.

In the current study a DNN was implemented as ABFT augmented matrix arithmetic as in [21]. The checksums of the weights are pre-computed while the with checksums are added to the inputs on-the-fly. As shown in Fig. 3, the checksum of the output vector after each matrix-matrix multiplication or addition is inspected to verify the correctness of the operations [21].

III. Experiments and Results

Both solutions were investigated with a DNN architecture [24] trained using the MNIST data-set [25]. The solutions were synthesized on a Xilinx Zynq-7000 System-on-Chip (SoC) [26] with clock frequency set to 200MHz, while the operating voltage of the FPGA section of the SoC was dynamically adjusted using Power-Management-BUS (PMBUS) commands from the host PC [27] according to detected computational errors. Firmware was written for the ARM processor in the SoC to control the FPGA acting as a neural accelerator. The processor feeds the network with input vectors and fetches the output via AXI ports. [28]. For further investigations on accuracy loss, fault model of the FPGA with voltage scaling may extracted to study larger model behavior with injected faults. The RTL of this model was generated using tools provided in [28].

After deployment of each solution on the FPGA, the supply voltage of the FPGA was gradually reduced by 10mV steps from the default 1.0V.

A. Fault detection through Self-supervised learning

The DNN model was modified slightly and re-trained with rotated images where prediction of input image rotation was considered as the pre-text task. After training accuracy of the main and the pre-text tasks were 78% and 93%, respectively.

The accuracy of the model for the main and the pre-text tasks versus voltage are plotted in Fig. 4 when the voltage is down-scaled close to 0.7V from 1.0V. A strong correlation between the accuracies of both tasks can be observed. Although SSL is not accurate in the detection of Point-of-First-Failure (POFF), it is able to detect failure regardless of the source of errors. The power consumption of the NN on the FPGA side is reported in Fig. 5, where the error detection points of both ABFT and the SSL techniques are recorded.

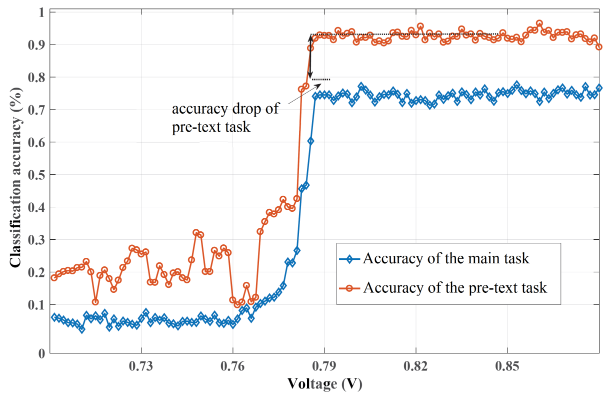


Figure 4. Voltage scaling impact on the accuracy of the main and pre-text tasks.

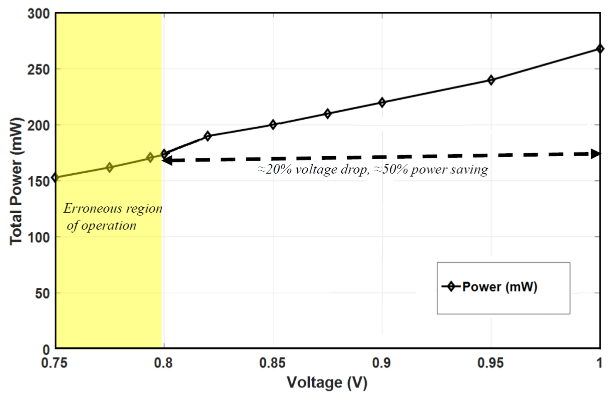


Figure 5. The impact of Voltage scaling on power dissipation.

B. Error detection through ABFT

The original trained model was converted into input to High-Level-Synthesis (HLS) and ABFT functionality was integrated. The HLS code of the network was modified in a manner that inputs to each layer are augmented with the checksum property and at the output the checksums are inspected as shown in Fig. 6. An accumulator at the input of multiplication/addition part of each layer augments the checksums. Likewise, at the output an accumulator together with a comparator verifies the integrity from ABFT checksums, and the results are sent to the control processor. The tool chain [28] was modified to automatically generate the necessary RTL code to integrate ABFT into the neural model computations. In addition to network output, the results of the checksum inspection blocks are routed back to the processor to report possible detections during the inference phase as single value output through AXI interfaces [26].

As analysed in [18], ABFT provides for high confidence, i.e., close to 100%, in detecting errors in neural computations. While it is not protecting the look-up-table based non-linear operations in the activation layers of the neural net, the delay-paths of multiplication and addition circuits are far longer and more likely to suffer from lower voltage induced phenomena. Figure 8 depicts the POFF for different operating points.

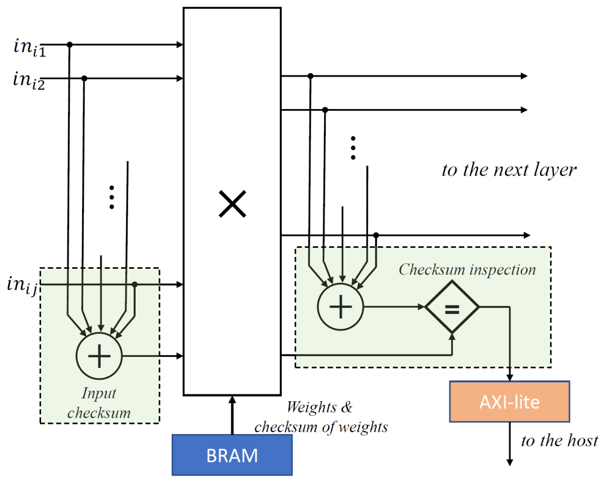


Figure 6. ABFT is added to the a multiplication section of a layer of neural network layer.

ABFT provides for high confidence in POFF detection as shown in Fig. 7 and Fig. 5, where, ABFT reported errors per 10k inferences are plotted together with the classification accuracy of the model. In all case, the ABFT detected errors in computations resulting in "no" performance degradation of the DNN model.

This enables substantial power savings by protecting the correctness of arithmetic operations of the DNN. With the employed model the ABFT overheads were less than 8%, while it would be even less with larger networks [21]. The overheads in power consumption are in the same ball park, while the power saving greatly outweigh the overheads. Unlike SSL, ABFT serves as an earlier detector, alarming just before the accuracy of the model would drop.

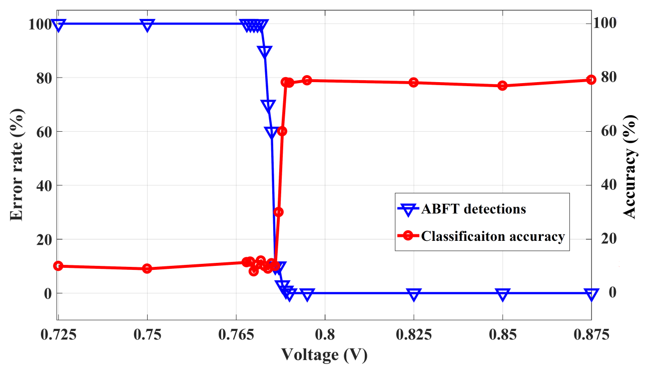


Figure 7. Impact of voltage scaling on an ABFT augmented DNN.

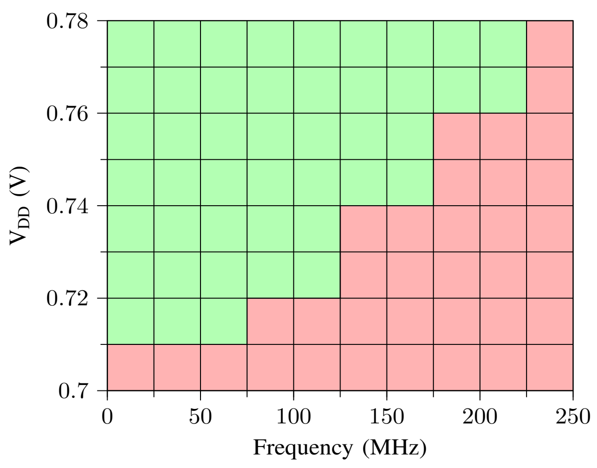


Figure 8. Shmoo plot for DNN deployed on Zynq FPGA. Appearance of ABFT detected error in computations is considered a fault.

IV. Comparison and Discussion

Jiang et al [12] and Salami et al. [29] propose to leverage voltage down-scaling and to exploit inherent fault-tolerance of neural networks for increasing the energy efficiency of DNN inference. However, those solutions reduce voltage without consideration to the associated dwindling accuracy of the network. In [30] extensive hardware simulation and modifications are required to augment the design with TEDs. Since, TEDs are not generally implementable for hard-blocks of FPGAs, their use is limited to ASIC implementation. The method achieves up to 3x improvement in energy efficient by reducing voltage down to near-threshold region. However due to high overheads a trade-off in adding TEDs was made amounting in a 3% accuracy drop. In [14] correlated predictions for consecutive frame in video processing were exploited. While the scheme does not incur any overheads, 20% of the faults escape detection.

The proposed solutions in this paper, are fully implementable in software. Their high reliability has been demonstrated, and the approaches are applicable to all types of neural network models.

The used self-supervised scheme is simplistic one. In the future more robust self-supervised learning schemes can be explored. Furthermore, since ABFT does not provide protection against faults in activation and pooling layers, it can benefit from co-use with the SSL. In addition to low-power applications, we envision utility from the proposed approaches for fault-critical systems [8].

Considering extensive utility of DNNs in 5G / 6G telecom [1] as the largest application area with a few billion mobile phones in service, the proposed solutions are easily portable to this area to curb growing challenges from limited battery-life of mobile phones to thermal issues in base-stations.

V. Conclusion

In this paper two solutions were investigated that enable safe voltage down-scaling of neural networks used for inference. With the SSL based scheme a pre-text task acts as a "canary bird in cage" indicator. The approach may incur some accuracy loss. The ABFT approach requires augmentation of neural computing with with checksums, and provides for high confidence in computational error detection. Both approaches add negligible overheads.

Detection of faults enabled conserving half of energy consumption on an FPGA through voltage reductions.

Acknowledgment

Part of this work was supported by the 6G Flagship Research Programme under Academy of Finland Grant 318927. Support from the Walter Ahlström foundation and Riitta ja Jarmo Takasen foundation is acknowledged by the first author.

References

[1] G. Paulin, F. Conti, L. Cavigelli, and L. Benini, "Vau da muntanialas: Energy-efficient multi-die scalable acceleration of rnn inference," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2021.

[2] G. Paulin, R. Andri, F. Conti, and L. Benini, "Rnn-based radio resource management on multicore risc-v accelerator architectures," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 9, pp. 1624–1637, 2021.

[3] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in NLP," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 3645–3650.

[4] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green AI," *Communications of the ACM*, vol. 63, no. 12, pp. 54–63, 2020.

[5] M. Safarpour, I. Hautala, M. B. Lopez, and O. Silven, "Transport triggered array processor for vision applications," in *International Conference on Embedded Computer Systems*. Springer, 2019, pp. 361–372.

[6] J. IJzerman, T. Viitanen, P. Jääskeläinen, H. Kultala, L. Lehtonen, M. Peemen, H. Corporaal, and J. Takala, "Aivotta: an energy efficient programmable accelerator for cnn-based object recognition," in *Proceedings of the 18th International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation*, 2018, pp. 28–37.

[7] H. Nakahara, Z. Que, and W. Luk, "High-throughput convolutional neural network on an fpga by customized jpeg compression," in *2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 2020, pp. 1–9.

[8] M. Sabih, F. Hannig, and J. Teich, "Fault-tolerant low-precision dnns using explainable ai," in *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. IEEE, 2021, pp. 166–174.

[9] Y. Du, X. Shang, and W. Shan, "An energy-efficient time-domain binary neural network accelerator with error-detection in 28nm cmos," in *2020 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*. IEEE, 2020, pp. 70–73.

[10] P. Pandey, P. Basu, K. Chakraborty, and S. Roy, "Greentpu: Predictive design paradigm for improving timing error resilience of a near-threshold tensor processing unit," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 7, pp. 1557–1566, 2020.

[11] Y. Lin, S. Zhang, and N. R. Shanbhag, "Variation-tolerant architectures for convolutional neural networks in the near threshold voltage regime," in *2016 IEEE international workshop on signal processing systems (SiPS)*. IEEE, 2016, pp. 17–22.

[12] W. Jiang, H. Yu, J. Zhang, J. Wu, S. Luo, and Y. Ha, "Optimizing energy efficiency of cnn-based object detection with dynamic voltage and frequency scaling," *Journal of Semiconductors*, vol. 41, no. 2, p. 022406, 2020.

[13] J. Kosaian and K. Rashmi, "Arithmetic-intensity-guided fault tolerance for neural network inference on gpus," *arXiv preprint arXiv:2104.09455*, 2021.

[14] L. K. Draghetti, F. F. dos Santos, L. Carro, and P. Rech, "Detecting errors in convolutional neural networks using inter frame spatio-temporal correlation," in *2019 IEEE 25th International Symposium on On-Line Testing and Robust System Design (IOLTS)*. IEEE, 2019, pp. 310–315.

[15] A. Mahmoud, N. Aggarwal, A. Nobbe, J. R. S. Vicarte, S. V. Adve, C. W. Fletcher, I. Frosio, and S. K. S. Hari, "Pytorchfi: A runtime perturbation tool for dnns," in *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. IEEE, 2020, pp. 25–31.

[16] X. Zhai, A. Oliver, A. Kolesnikov, and L. Beyer, "S4l: Self-supervised semi-supervised learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1476–1485.

[17] K.-H. Huang and J. A. Abraham, "Algorithm-based fault tolerance for matrix operations," *IEEE transactions on computers*, vol. 100, no. 6, pp. 518–528, 1984.

[18] M. Safarpour, R. Inanlou, and O. Silvén, "Algorithm level error detection in low voltage systolic array," *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2021.

[19] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," *arXiv preprint arXiv:1803.07728*, 2018.

[20] D. Hendrycks, M. Mazeika, S. Kadavath, and D. Song, "Using self-supervised learning can improve model robustness and uncertainty," *arXiv preprint arXiv:1906.12340*, 2019.

[21] K. Zhao, S. Di, S. Li, X. Liang, Y. Zhai, J. Chen, K. Ouyang, F. Cappello, and Z. Chen, "Ft-cnn: Algorithm-based fault tolerance for convolutional neural networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1677–1689, 2020.

[22] M. Vijay and R. Mittal, "Algorithm-based fault tolerance: a review," *Microprocessors and Microsystems*, vol. 21, no. 3, pp. 151–161, 1997.

[23] M. Safarpour, L. Xun, G. V. Merrett, and O. Silven, "A high-level approach for energy efficiency improvement of fpgas by voltage trimming," *TechRxiv*, 2021.

[24] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[25] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

[26] L. H. Crockett, R. Elliot, M. Enderwitz, and R. Stewart, *The Zynq book: embedded processing with the ARM Cortex-A9 on the Xilinx Zynq-7000 all programmable SoC*, 2014.

[27] A. F. Beldachi and J. L. Nunez-Yanez, "Accurate power control and monitoring in zynq boards," in *2014 24th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2014, pp. 1–4.

[28] R. Novickis, D. J. Justs, K. Ozols, and M. Greitāns, "An approach of feed-forward neural network throughput-optimized implementation in fpga," *Electronics*, vol. 9, no. 12, p. 2193, 2020.

[29] B. Salami, E. B. Onural, I. E. Yuksel, F. Koc, O. Ergin, A. C. Kestelman, O. Unsal, H. Sarbazi-Azad, and O. Mutlu, "An experimental study of reduced-voltage operation in modern fpgas for neural network acceleration," in *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2020, pp. 138–149.

[30] N. D. Gundi, T. Shabanian, P. Basu, P. Pandey, S. Roy, K. Chakraborty, and Z. Zhang, "Effort: Enhancing energy efficiency and error resilience of a near-threshold tensor processing unit," in *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2020, pp. 241–246.