Statistical Modeling of Soft Error Influence on Neural Networks

Haitong Huang, Xinghua Xue, Cheng Liu, Member, IEEE, Ying Wang, Member, IEEE, Tao Luo, Long Cheng, Huawei Li, Senior Member, IEEE, and Xiaowei Li, Senior Member, IEEE

Abstract—Soft errors in large VLSI circuits have a significant impact on computing- and memory-intensive neural network (NN) processing. Understanding the influence of soft errors on NNs is critical to protect against soft errors for reliable NN processing. Prior work mainly relies on fault simulation to analyze the influence of soft errors on NN processing. They are accurate but usually specific to limited configurations of errors and NN models due to the prohibitively slow simulation speed especially for large NN models and datasets. With the observation that the influence of soft errors propagates across a large number of neurons and accumulates as well, we propose to characterize the soft error-induced data disturbance on each neuron with a normal distribution model using the central limit theorem and develop a series of statistical models to analyze the behavior of NN models under soft errors in general. The statistical models reveal not only the correlation between soft errors and the accuracy of NN models but also how NN parameters, such as quantization and architecture affect the reliability of NNs. The proposed models are compared with fault simulations and verified comprehensively. In addition, we observe that the statistical models that characterize the soft error influence can also be utilized to predict fault simulation results in many cases and we explore the use of the proposed statistical models to accelerate fault simulations of NNs. Our experiments show that the proposed accelerated fault simulation provides almost two orders of magnitude speedup with negligible loss of simulation accuracy compared to the baseline fault simulations.

Index Terms—Fault analysis, fault simulation, neural network (NN) reliability, statistical fault modeling.

Manuscript received 17 September 2022; revised 26 January 2023; accepted 5 April 2023. Date of publication 11 April 2023; date of current version 20 October 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 62174162, in part by the National Key Research and Development Program of China under Grant 2022YFB450040; and in part by the Singapore Government’s Research, Innovation and Enterprise 2020 Plan (Advanced Manufacturing and Engineering Domain) under Grant A16878003. This article was recommended by Associate Editor Y. Li. (Corresponding author: Haitong Huang.)

Haitong Huang, Xinghua Xue, Cheng Liu, Ying Wang, and Xiaowei Li are with the State Key Laboratory of Processors, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China, and also with the Department of Computer Science, University of Chinese Academy of Sciences, Beijing 100190, China (e-mail: huanght@ict.ac.cn).

Tao Luo is with the Institute of High Performance Computing, A*STAR, Singapore 138632.

Long Cheng is with the School of Control and Computer Engineering, North China Electric Power University, Beijing 102206, China.

Huawei Li is with the State Key Laboratory of Processors, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China, and also with Peng Cheng Laboratory, Shenzhen 518055, China.

Digital Object Identifier 10.1109/TCAD.2023.3266405

I. INTRODUCTION

RECENT years have witnessed the widespread adoption of neural networks (NNs) in various applications [1]. Many of the applications, such as autonomous driving, medical diagnosis, and robot-assisted surgery, are safety critical [2], [3], [4]. Failures in these applications can cause threats to human life and dramatic property loss. To ensure application safety, the reliability of NN accelerators, which are increasingly utilized for their competitive advantages in terms of performance and energy efficiency [5], [6], must be comprehensively evaluated and verified.

With the continuously shrinking semiconductor feature sizes and growing transistor density, the influence of soft errors on large-scale chip designs becomes inevitable [7], [8]. A variety of analysis work have been conducted to investigate the influence of soft errors on NN execution reliability from distinct angles recently [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20]. For instance, Reagen et al. [9] investigated the relationship between fault error rate and model accuracy from the perspective of models, layers, and structures. Li et al. [10] experimentally evaluated the resilience characteristics of deep NN systems (i.e., NN models running on customized accelerators) and particularly studied the influence of data types, values, data reuses, and types of layers on NN resilience under soft errors, which further inspires two efficient protection techniques against soft errors. He et al. [19] had the major NN accelerator architectural parameters considered to obtain more accurate fault analysis. Xu et al. [11], [12] explored the influence of persistent faults on FPGA-based NN accelerators with hardware emulation.

Despite the efforts, they mainly rely on a large number of fault simulations on either software or FPGAs with limited fault injection configurations. The simulation-based analysis is relatively accurate on specific NN models and fault configurations, but the simulation remains rather limited compared to the entire large design space. Hence, there is still a lack of generality using the simulation-based fault analysis. In fact, some of the simulation results can lead to contradictory conclusions. For instance, the experiment results in Ares [9] demonstrate that the model accuracy of typical NNs drops sharply when the bit error rate (BER) reaches $1 \times 10^{-7}$ while the experiments in [21] reveal that the model accuracy starts to drop when the BER is larger than $1 \times 10^{-5}$. In fact, both experiments are correct and the difference is mainly caused by the different quantization setups. Although this can be fixed with...
more comprehensive fault simulations, the total number of fault simulations can increase dramatically given more analysis factors, which will lead to rather expensive simulation overhead accordingly. Hence, more general analysis approaches are demanded to gain sufficient understanding of the influence of soft errors on NNs.

Moreover, the simulation-based fault analysis can be extremely time consuming under real-world applications with large NNs and datasets, which hinders its use in practice. Take NN vulnerability analysis that locates the most fragile part of an NN to facilitate selectively protection against soft errors as an example. Suppose we need to select the most fragile $k$ layers of an NN with $N$ layers. A straightforward simulation-based approach needs to conduct $C^k_N$ experiments on the target test dataset. When $N = 153$, $K = 5$, $C^5_{153} = 654,045,930$. Assume the NN fault simulation speed is 50 frame per second (fps) and 1000 samples are required for accuracy estimation. The evaluation of a single configuration takes 20 s and the full evaluation takes around 420 years, which generally can not be afforded. As a result, many existing solutions [13] can only adopt heuristic algorithms to address this problem approximately.

In order to achieve more efficient fault analysis of soft errors on NN execution, we develop a statistical model to gain insight of the influence of soft errors on NN models with much less experiments or even no experiments. The basic idea is to view the soft errors randomly distributed across the NN processing via statistical analysis and investigate the influence of soft errors on the NN model accuracy. Specifically, we utilize a normal distribution model to characterize the distribution of the neurons in NNs according to central limit theorem and analyze the computing error distribution induced by the random soft errors first. On top of the models, we further investigate the influence of NN depth, quantization, classification complexity on the resilience of NNs under soft errors. At the same time, we verify the proposed modeling and analysis with fault simulation. Finally, we further leverage the statistical models to accelerate the time-consuming fault simulation by performing fault analysis with intermediate data rather than model accuracy directly.

The contributions of this work can be summarized as follows.

1) We propose a series of statistical models to characterize the influence of soft errors on NN processing for the first time. The models enable relatively general analysis of NN model resilience under soft errors.

2) We leverage the statistical models to investigate how the major NN parameters, such as quantization, number of layers, and number of classification types affect the NN resilience, which can be utilized to guide the fault-tolerant NN design.

3) With the proposed statistical models, we can also accelerate conventional fault simulation of NN processing under soft errors by almost two orders of magnitude through simplifying the fault injection and replacing model accuracy analysis with more cost-effective intermediate parameter analysis.

4) We validate the proposed model-based soft error influence analysis of NNs and demonstrate significant fault simulation acceleration with comprehensive experiments.

The remainder of this article is organized as follows. Section II briefly introduces prior fault analysis of NN models. Section III illustrates the proposed statistical models for NN reliability analysis under soft errors. Section IV presents the use of the proposed statistical models to characterize the influence of NN parameters on NN resilience over soft errors. Section V mainly demonstrates how the proposed statistical models can be utilized to accelerate the fault simulation of NNs under soft errors. Section VI concludes this article.

II. RELATED WORK

Fault simulation is key to understand the influence of hardware faults on the NN processing and is the basis for fault-tolerant NN model and accelerator designs [19], [20], [22], [23], [24] in various application scenarios. For instance, fault simulations in [13] and [25] are utilized to investigate the vulnerability of NNs and accelerators, which enables selectively hardware protection against various hardware faults with minimum overhead. Fault simulations in [9], [26], and [27] are applied to investigate the design tradeoffs between model accuracy loss and computing errors, which can be leveraged for energy-efficient NN accelerator design through approximate computing and voltage scaling. Hence, a variety of fault simulation work have been developed in the past few years [22], [23], [25], [28], [29], [30], [31], [32], [33], [34]. They can generally be divided into two categories depending on the fault simulation abstraction layers.

First, neuronwise fault simulation that injects faults to neurons or weights are mostly widely adopted in prior works [9], [10], [22], [23], [25], [27], [28], [29], [30], [31], [32] and have been verified according to [9]. Although faults are originated from the underlying computing engines, these simulation frameworks typically adopt abstract bit-flip or stuck-at faults and include little hardware architecture details. To further improve the fault analysis precision, Xue et al. [13] developed an operation-level fault analysis framework such that hardware faults are injected to basic operations, such as multiplication and accumulation, which is utilized to explore the influence of winograd convolution on resilience of NN processing. He et al. [19] had NN accelerator architectural parameters combined with high-level simulation of NN processing with transient faults to achieve both high-fidelity and high-speed resilience study of general NN accelerators. In summary, the above fault simulation work are mostly built on existing deep learning frameworks, such as PyTorch and TensorFlow and are flexible for various fault simulations while the parallel processing capability can be negatively affected by the low-level fault injection substantially. To this end, BinFI [33] mixed the analysis model in error injection, and utilized binary search algorithm to speed up the error injection. There have been efforts made to understand artificial NNs from statistical perspectives [35], [36] and characterize soft error rate with...
statistical models [37], [38], which demonstrated the potential of using statistical models for NNs and soft errors. However, there is still a lack of analytical work on investigating the influence of soft errors on convolutional NNs so far.

The other category is circuit-layer fault simulation that typically conducts fault simulation on circuit designs at either gate level or RTL level. It is already well supported by commercial EDA tools like TetraMAX and can achieve high simulation precision, but it can be extremely slow for NN accelerators that include a larger number of transistors. An alternative approach is fault emulation that conducts fault simulation on FPGAs [11], [12], [39], [40]. Similarly, NVIDIA SASSIFI [24] developed a fault injection mechanism for GPU and can be utilized for rapid fault analysis of NNs on GPUs. Basically, these fault simulation frameworks greatly improve the fault simulation speed but rely on specific hardware prototypes and architectures which are usually difficult to scale and modify.

Despite the efforts, simulation-based fault analysis is mainly applicable to specific setups in terms of NN models, target hardware architectures, and fault configurations. A comprehensive fault analysis requires a huge number of fault simulations as discussed in Section I which is prohibitively expensive. As a result, the fault analysis generality is usually limited. In fact, because of the limited fault simulation setups, some of the simulation-based fault analysis even produces inconsistent results. For instance, Reagen et al. [9] concluded that the resilience of different layers of the NN may vary up to 2781×. Nevertheless, Banerjee et al. [41] revealed a different conclusion based on the Bayesia fault injection and analysis. The problem poses significant demands for more general and faster fault analysis.

III. SOFT ERROR INDUCED NEURAL NETWORK COMPUTING ERROR MODELING

In this work, we mainly analyze the influence of soft errors on NN processing with modeling to gain insight of the NN fault tolerance and guide the fault-tolerant design of NN models and accelerators. Soft errors induced computing errors propagate rapidly across layers of NNs and the influence of the different soft errors is accumulated on neurons of the NN. Basically, the influence of random soft errors are distributed and accumulated on a large number of neurons. Hence, it can be characterized with a normal distribution model according to central limit theorem. With the distribution model, we can further estimate the NN outputs and the model accuracy loss eventually, which can be fast and general as well.

A. Model Notations

NNs can be considered as multilayer nonlinear transformation and the transformation in a layer \( l \) can be formulated as (1) where \( f_i \) represents the transformation operation, \( x_i \) represents the input activations, \( x_{i+1} \) represents the output activations. Particularly, for convolutional NNs, \( w_i \) represents weights in layer \( l \), \( * \) denotes convolution or full connection, \( b_i \) is the bias, and \( \varphi \) represents a nonlinear activation function

\[
x_{i+1} = f_i(x_i) = \varphi(x_i \ast w_i + b_i).
\]

While soft errors may happen in any layer of an NN and propagate across the NN layers, we utilize (2) to characterize the relation between input activations in layer \( l \) and the output activations of the layer that is \( m \) layers behind to facilitate the fault analysis. Note that \( x_l \) denotes input activations in layer \( l \) and \( F^{l+m}_l(x_l) \) denotes output activations of layer \( l + m \)

\[
F^{l+m}_l(x_l) = f_i(m-1)(\cdots f_i(1)f_i(x_l)).
\]

Soft errors propagate along with layers of the NN and can cause input variation on all the following layers. Suppose bit flip errors occur in weights or output activations at the \((l-1)\)th layer. The induced variation at the \( l \)th layer is denoted as \( \delta_l \) and the variation at the \((l+\alpha)\)th layer is denoted as \( \Delta^\alpha_l \).

These variation can be calculated with (3). For the variation of the overall NN outputs induced by soft errors in layer \( l \), we denote it as \( \Delta^N_l \) where \( N \) refers to the total number of layers in the NN and the notation is simplified as \( \Delta_l \) in the rest of this article

\[
\Delta^\alpha_l = x^\prime_{l+1} - x_{l+1} = f_i(x_l + \delta_l) - f_i(x_l)
\]

\[
\Delta^\alpha_l = x^\prime_{l+2} - x_{l+2} = F^{l+1}_{l+1}(x_l + \delta_{l+1}) - F^{l+1}_{l+1}(x_l)
\]

\[
\Delta^\alpha_l = x^\prime_{l+m} = F^{l+m}_l(x_l + \delta_{l+m}) - F^{l+m}_l(x_l).
\]

To quantize the soft error induced computing variation of the NN, we utilized RMSE \( l = \|\Delta/\mu\|_2 = \sqrt{\text{var}(\Delta)} \) as a metric initially where \( \mu \) is the vector length of the NN output. However, RMSE is sensitive to the data range of activations that may vary over different layers of the same NN, so we have RMSE further normalized and utilize RMSE Ratio (RRMSE) \( \text{RRMSE}_l = \text{RMSE}_l/\sqrt{\text{var}(x_l)} \) instead. The metric is more convenient to calculate compared to the model accuracy that relies on statistical results of a large number of samplings. The correlation between RRMSE and model accuracy will be illustrated in the rest of this section.

B. Assumptions and Lemmas

Recent work [42], [43] already demonstrated that the distribution of weights in NNs fits well with t-Location Scale distribution which is a long-tail normal distribution. While output activations are generally accumulation of many weighted input activations. Suppose the input activations are random variables. Then, the output activations will be close to a normal distribution according to central limit theorem. Particularly, activations are usually close to zero to make full use of the nonlinear activation function. Similarly, fully propagated activation errors are also accumulation of multiple random errors propagated from neurons in upstream layers and belongs to a normal distribution. In summary, weights, activations of the NN, and activation errors can all be approximated to normal distribution centered at zero and they can be formulated as follows:

\[
w_l \sim N(0, \text{var}(w_l))
\]

\[
x_l \sim N(0, \text{var}(x_l))
\]

\[
\Delta_l \sim N(0, \text{var}(\Delta_l)).
\]
the error distribution. Note that a single bit error is injected to an input activation of models. m, s, and k stand for mean, skewness, and e-kurtosis. They are 0 on standard normal distribution. The experiment result is shown in Fig. 2. It can be observed that the neuron errors generally fit well with a normal distribution model centered at zero except that in Conv2 in which the input errors have not propagated in a layer of the NN

\[
\Delta_{l+1}^{(j)} = \left( (x_l + \Delta_l) \ast w_l - x_l \ast w_l \right)^{(j)} = \sum_{i=1}^{m} \Delta_i^{(j)} \ast w_l^{(j)}.
\]

C. Bit Flip Influence Modeling

To understand the influence of bit flip soft errors on NN processing, we start to investigate the influence of bit flip on a single data. Take an activation \( x \) quantized with int8 as an example, the quantized activation \( x_Q \) can be represented with (8) where bound represents the dynamic range of activations in a layer of the NN

\[
x_Q = \left\lfloor \frac{128 \cdot x}{\text{bound}} \right\rfloor.
\]
variation caused by soft errors. Suppose the range of an activation is \([-\text{bound}, \text{bound}\)]\), a bit flip on most significant bit (MSB) results in a change of \(\pm\text{bound}\). Similarly, a bit flip on the following bit leads to a change of \(\pm\text{bound}/2\). In this case, the expected change of data i.e., \(\sigma_d\) quantized with int8 and int16 given a random bit flip can be represented with (9) and (10), respectively. When we double the \(\text{bound}, \sigma_d\) doubles. When the quantization data width doubles, \(\sigma_d\) shrinks by \(\sqrt{2}\)

\[
\sigma_{8_{\text{int}}} = \sqrt{\frac{7}{8} \sum_{\text{bit}=0}^{7} \left(\frac{\text{bound}}{2^{\text{bit}}}\right)^2} \approx \frac{\text{bound}}{\sqrt{6}} \tag{9}
\]

\[
\sigma_{16_{\text{int}}} = \sqrt{\frac{15}{16} \sum_{\text{bit}=0}^{15} \left(\frac{\text{bound}}{2^{\text{bit}}}\right)^2} \approx \frac{\text{bound}}{\sqrt{12}}. \tag{10}
\]

On top of the bit error influence of a single data, we scale the analysis to estimate the output data variation of a convolution operation induced by soft errors injected to either weights or activations. Suppose \(ic\) refers to the number of input channel, \(oc\) refers to the number of output channel, \(H \times H\) stands for the size of a feature map, and \(K \times K\) refers to the size of a convolution kernel.

**Disturbance in Weights:** Disturbance in a single weight affects output activations of an entire channel, i.e., \(H^2\) activations. According to definition \(\text{RMSE} = \|\Delta/n\|\) and Lemma 2 in Section III-B, the average variation of weights i.e., \(\text{var}(w)\) depends on the amount of input activations \(m = K^2 \times ic\). \(\text{var}(w) = \frac{1}{m} \text{var}(x_{i+1})/\text{var}(x_i) \approx 1/m = 1/(K^2 \times ic)\). So the average computing variation of output activations can be calculated with (11)

\[
\text{RMSE_w} = \frac{\|\Delta/n\|}{2} = \sqrt{\frac{H^2 \times \text{var}(w)}{H^2 \times oc}} = \frac{\sigma_d \sqrt{\text{var}(w)}}{\sigma_d} \approx \frac{\text{bound}}{K \sqrt{ic \times oc}}. \tag{11}
\]

**Disturbance in Activations:** Suppose the average value of an activation in a layer is \(\sigma_a\). Disturbance in a single activation will affect a window \((K^2)\) of output activations in all the \(oc\) channels, i.e., \(oc \times K^2\) activations. Suppose the feature map size is \(H \times H\), then the average influence of the disturbance on an output activation can be calculated with

\[
\text{RMSE_a} = \frac{\|\Delta/n\|}{2} = \sqrt{\frac{oc \times K^2 \times \sigma_a^2 \text{var}(w)}{H^2 \times oc}} = \frac{\sigma_a}{H \sqrt{ic}}. \tag{12}
\]

Since the variation of activations and weights is generally constant given a specific model and data set according to the assumptions in Section III-B, \(\text{RRMSE}_i = \text{RMSE}_i/\sqrt{\text{var}(x_i)}\) is consistent with \(\text{RMSE}\) accordingly.

**D. Relation Between RRMSE and Classification Accuracy**

Since the model accuracy of a typical classification task is based on statistics of a number of classification tasks and it is difficult to formulate with NN computing directly, we utilize \(\text{RRMSE}\) defined in Section III-A as an alternative metric to measure the influence of soft errors on model accuracy metric which is more closely related with NN computing.

To begin, we start with a simple binary classification task and it includes only two output neurons in the last layer of the network. Then, the classification depends on larger output neuron. Suppose the two output neurons are \(y(0)\) and \(y(1)\), and assume \(y(0) > y(1)\) without loss of generality. When there are random errors injected to the NN, the two output neurons follow normal distribution accordingly and they can be formulated with \(Y(0) \sim N(y(0), \text{var}(\Delta_0)), Y(1) \sim N(y(1), \text{var}(\Delta_1))\) according to Section III-B. In this case, the probability of wrong classification is essentially that of \(Y(0) < Y(1)\). The distribution of \(Y(0) < Y(1)\) can be formulated with (13), which is a shifted and scaled normal distribution model. Hence, the probability when \(Y(0) - Y(1) > 0\) can be calculated with (14) where normcdf stands for cumulative distribution function of normal distribution

\[
y(0) - y(1) \sim N(y(0) - y(1), 2\text{var}(\Delta_1)) \tag{13}
\]

\[
\text{Acc} = \text{normcdf}\left(\frac{y(0) - y(1)}{2\text{var}(\Delta_1)}\right) + \frac{1}{2}. \tag{14}
\]

While \(\text{var}(y) = (1/4)(y(0) - y(1))^2\) in a typical binary classification task, (14) can be converted to (15) according to the definition of \(\text{RMSE}\) and \(\text{RRMSE}\) in Section III-A

\[
\text{Acc} = \frac{1}{2} \text{erf}\left(\frac{2\text{var}(y)}{2\text{RRMSE}_y}\right) + \frac{1}{2} \tag{15}
\]

For multiclass classification models, suppose there are \(nc\) classification types and the corresponding outputs of the classification model are denoted as \(y(i), i = 0, 1, 2, \ldots, nc - 1\). Assume the expected output is \(y(0)\) which is larger than any other output \(y(i), i = 1, 2, \ldots, nc - 1\). Similar to the binary classification problem, a classification error happens when any of the outputs \(y(i), i = 1, 2, \ldots, nc - 1\) is larger than \(y(0)\) given Gaussian disturbance i.e., \(\text{RRMSE}\) according to the definition of \(\text{RMSE}\). In this case, the model accuracy subjected to errors is the integration of the multiplicity of the probability density function of \(y(0)\) and the cumulative distribution function of the rest outputs \(y(i), i = 1, 2, \ldots, nc - 1\). It can be calculated with (16), shown at the bottom of the page.

Since the analytical model is usually difficult to calculate, we replace it with an empirical model as shown in (17) shown at the bottom of the next page. It is essentially an sigmoid function variant and has two parameters i.e., \(m\) and \(s\) included. When there are no errors, the output of (17) is the accuracy
of a clean NN model. When there are too many errors, the output of (17) becomes $1/nc$ which represents classification accuracy of random guessing. The empirical model can be determined given very few sampling data points of RRMSE and the corresponding classification accuracy.

To verify the proposed models of correlation between RRMSE and NN classification accuracy, we take VGGNet-11 and ResNet-18 on ImageNet as typical NN examples to compare the analytical models and empirical models to the ground truth results. Particularly, we have two different evaluation approaches performed for the model comparison. In the first approach, we have a single true-positive images from ImageNet utilized and repeat the execution for 10,000 times on each error injection rate setup. It is denoted as single image analysis and it removes the influence of image variations from the analysis. In the second approach, we have 10,000 different images randomly selected from ImageNet and evaluated for each error injection rate setup. It is denoted as multiple image analysis and it has the image variations incorporated in the analysis. For the soft error injection, we have 31 different BER setups ranging from 1E-7 to 1E-4 conducted in the experiment. Ground truth of RRMSE and classification accuracy can be obtained from experiments directly. Analytical model can be determined given the RRMSE and $nc$ while the empirical model can be determined with fitting on 4 data points evenly selected from ground truth data. The comparison is shown in Fig. 3. It can be observed that the proposed analytical model is close to the ground truth data in the single image analysis setup but it has variation under multiple image analysis setup. This is expected as the proposed analytical model fails to characterize the influence of image variations. In contrast, empirical model fits much better on both single image analysis and multiple image analysis despite the lack of explainability. Nevertheless, it is clear that the model accuracy decreases monotonically with the increase of RRMSE, which allows us to characterize the influence of soft errors with RRMSE which can be obtained more conveniently compared to model accuracy.

### E. Error Influence Aggregation

In this section, we mainly explore how the influence of different errors aggregate on the same output neurons. Suppose RRMSE represents RRMSE of an NN output, it is the accumulation of $n$ independent random errors propagated from different layers. As mentioned, the influence of the random errors follows normal distribution model. The accumulation of these independent random errors can be formulated with (18)

$$
RRMSE = \sqrt{\sum_{i=1}^{L} \sum_{l=1}^{n_l} RRMSE_{(l,i)}^2}
$$

where $RRMSE_{(l,i)}$ denotes the RRMSE induced by a neuron fault on layer $l$. As the number of neurons in an NN is extremely large, it remains taxing consuming and inefficient to conduct neuronwise fault analysis. To address the problem, we have (18) further converted to layerwise fault analysis where $RRMSE_{(l)}$ denotes RRMSE of NN output caused by all the errors in layer $l$ and $L$ denotes the total number of NN layers

To verify the error aggregation model, we conduct layerwise fault injection on VGGNet-11 and Resnet-18 quantized with int8 at different error injection rate BER $\in [0, 1e^{-5}, 2e^{-5}]$. Then, we randomly select 32 different combination of layerwise error injection configurations and compare the ground truth RRMSE with that estimated with (18). The total number of bit errors varies from $10^1$ to $10^3$ under this setting. The comparison shown in Fig. 4 reveals that the error aggregation model fits well with the simulation results when a small number of error combinations occur simultaneously, but clear deviation can be observed when the number of errors gets

$$
\text{Acc}(\text{RRMSE}) = \int_{-\infty}^{\infty} \text{normpdf}(x) \cdot (\text{normcdf}(x + 1/\text{RRMSE}))^{nc-1} dx
$$

$$
= \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \cdot \left( \int_{-\infty}^{t} \frac{1}{\sqrt{2\pi}} e^{-\frac{(t+1)/\text{RRMSE})^2}{2}} dt \right)^{nc-1} dx
$$

$$
\text{Acc}(\text{RRMSE}) = \frac{(1 + e^{-ms})(\text{Acc}_{\text{clean}} - nc^{-1})}{1 + e^{m(\text{RRMSE} - ms)}} + nc^{-1}
$$

Fig. 3. Analytical model and empirical model comparison on both (a) single image analysis experiment and (b) multiple image analysis experiment. The accuracy obtained with fault injection, analytical model, and empirical model is presented on the left y-axis. The accuracy deviation of analytical model and empirical model over fault injection results are presented on right y-axis.
larger. This deviation may be caused by both the complex interaction among the different errors and the increased error distribution variations.

IV. MODELING FOR NEURAL NETWORK RESILIENCE ANALYSIS

With the above modeling of soft error influence on NN accuracy, we can further utilize the models to analyze the influence of the major NN design parameters on the resilience of NNs subjected to soft errors, which can provide more general analysis compared to simulation-based approaches. Specifically, we investigate influence of NN layers, quantization, and number of classification types, respectively, on NN resilience and they will be illustrated in detail in this section. In order to verify the models, we compare the model-based results with fault injection-based results. Specifically for the fault injection, we use 10,000 images randomly selected from ImageNet for the accuracy evaluation and inject random bit flip errors to all the neurons in an NN model which is consistent with prior fault injection frameworks including Ares [9] and PyTorchFI [22]. Although the bit flip errors in neurons are similar to bit flip errors in on-chip buffers, they are actually measured from the perspective of NN processing and do not target any specific hardware computing engines. Essentially, it takes the computing of each neuron as a basic operation and all the hardware faults used for the neuron computing are abstracted as bit flip errors at the output neuron. The bit error model for NN processing is partly validated in Ares [9] with voltage down-scaling of SRAM in an NN accelerator, though more systematic validation remains demanded. The problem is beyond the scope of this work.

A. Influence of Number of Layers on NN Resilience

Existing NNs mainly consist of convolution and full-connected layers that involve only multiply accumulation operations, nonlinear activation layer (e.g., ReLU) and linear transformation layer (e.g., batch normalization). According to Lemma 4 in Section III-B, the output error metric of the $l$th layer i.e., $\text{RRMSE}_{l}$ is roughly constant across convolution and full-connected layers. The batchnorm layer is essentially a linear operator that scales the error and activation equally, it does not directly affect RRMSE. As for the nonlinear activation functions, they will affect the outputs directly. For example, ReLU will filter out computing errors when activation is negative. Suppose the propagated errors distribute evenly on positive and negative activation values, it can be expected that ReLU will filter out half activations and errors simultaneously. Then, Lemmas 2 and 3 can approximately change to $\text{var}(x_{l+1}) \approx (m/2)\text{var}(x_l)\text{var}(w_l)$ and $\text{var}(\Delta_{l+1}) \approx (m/2)\text{var}(\Delta_l)\text{var}(w_l)$ if we merge the effect of $\Delta$. Similarly, ReLU in to it. Thereby, RRMSE remains unchanged. In summary, for typical NNs built with convolution-batchnorm-ReLU layers, NN depth will not affect RRMSE i.e., NN resilience in general. In order to verify the analysis, we have random bit flip errors injected to neurons in different convolution layers of VGGNet-11 and ResNet-18. The BER is set to be 1E-6 in this experiment when the model shows notable accuracy drop. Then, we evaluate RRMSE of the following layers of the NNs over the corresponding golden reference output. RRMSE of the NN layers is shown in Fig. 5. It can be seen that RRMSE on different layers varies in a small range in general for each specific fault injection despite the layer locations of the error injection. Basically, it demonstrates that the influence of soft errors remains steady across the different layers and NN depth will not affect the fault tolerance of the NN model in general. It is true that small variations rather than constant as estimated in Lemma 4 can be found in RRMSE of different layers in Section III-B. This is probably caused by the nonlinear operations that may not be ideal. Nevertheless, the influence of ReLU and batchnorm layers on RRMSE is generally small according to the experiments though they are not fully considered in the analytical models.

B. Influence of Quantization on NN Resilience

According to the analysis in Section III-D, we notice that quantization bound has straightforward influence on data variation induced by a single bit flip and affects RRMSE.
Fig. 6. Influence of quantization bound on RRMSE. (a) VGGNet-11. (b) ResNet-18.

Accordingly, NN models can choose different quantization bound with little accuracy loss in practice, so it can be expected that smaller quantization bound can improve the NN model reliability subjected to soft errors without accuracy penalty. Fig. 6 presents RRMSE of VGGNet-11 and ResNet-18 subjected to soft errors with various quantization bound while the BER is set to be 1E-6. It reveals that RRMSE that is closely related with the NN model accuracy as demonstrated in Section III-D increases almost linearly with the quantization bound experimented with all the different layers. On the other hand, we also observe that clean NN classification accuracy remains steady in a wide range of quantization bound setups, which indicates that it is possible to improve the NN reliability without accuracy penalty by choosing appropriate quantization bound.

According to the comparison in (9) and (10), we notice that the average disturbance induced by a single bit error for NN model quantized with int16 is $\sqrt{2} \times$ smaller than that quantized with int8. On the other hand, the expected total number of bit errors for model quantized with int16 is twice larger than that quantized with int8 given the same BER. According to (18), RRMSE of the same NN model with more bit errors will be $\sqrt{2} \times$ larger. Hence, RRMSE of an NN is equal to RRMSE of the same NN model with int8 and int16 as examples and conduct fault simulation to verify the model-based analysis. The BER is set to be 5E-6 and the quantization bound is set to be the same for both quantization data width. Due to the time-consuming fault injection simulation, we only have 1000 images used for each data-point in Fig. 7. The experiment results demonstrate that both RRMSE and accuracy of NN models are generally steady despite the quantization data width, which is consistent with the model-based analysis.

C. Influence of Classification Complexity on NN Resilience

Intuitively, we notice that easier deep learning tasks are generally more resilient to errors, but it is usually difficult to define the complexity of an NN processing task. Equation (16) provides a model to characterize the relation between the number of classification types and classification accuracy. When we take the number of classification types as a metric of NN complexity, it provides a simple yet efficient angle to characterize the relation between NN complexity and NN resilience. The model in (16) proves that NNs with more classification types are more vulnerable subjected to the same number of errors. To verify this, we take VGGNet-11 and ResNet-18 on ImageNet as examples and then configure them for a set of classification tasks with different number of classification types i.e., $nc$ ranging from 2 to 1000. Then, we explore the resulting accuracy of these classification tasks subjected to the same bit error setups. The experiment result is shown in Fig. 8. It reveals that NN models with less classification types generally have much higher accuracy and the accuracy drops slower with increasing BER compared to that with more classification types.

V. MODELING FOR FAULT SIMULATION ACCELERATION

Fault simulation is a typical practice for NN resilience analysis under hardware errors and several fault simulation tools [19], [22], [23] targeting at NNs have been developed with different tradeoffs between simulation accuracy and speed. Usually, a large number of errors need to be injected and a variety of fault configurations need to be explored to ensure steady simulation results, which can be rather time consuming and expensive. Orthogonal to prior fault simulation approaches, we mainly investigate how the modeling proposed in this work can be utilized to accelerate the fault simulation with negligible simulation accuracy loss. The experiment
configuration including dataset and quantization is the same as that described in Section IV.

A. Fault Simulation Acceleration Approaches

To begin, we will introduce three statistical model-based approaches that can be utilized to accelerate general fault simulation. The basic idea is to leverage the proposed statistical models to predict the fault simulation results with only a fraction of simulation setups and reduce the number of fault simulation of all the possible fault configurations. The three acceleration approaches are listed as follows.

First, according to (17), we can characterize the relation between RRMSE and model accuracy under various BER setups with only four data points. Moreover, RRMSE of a model can be obtained with less input images and converges much faster than that of model accuracy. Particularly, we take VGGNet-11 on ImageNet as an example and investigate how the RRMSE and model accuracy of VGGNet-11 changes with different number of input image samples given the same BER. The BER is set to be 1E-6. The experiment result in Fig. 9 confirms the advantage of using RRMSE in terms of convergence. With both the RRMSE and the correlation curve of RRMSE and model accuracy, we can obtain the correlation curve of BER and model accuracy much faster compared to conventional fault simulation experiments.

Second, with the analysis in Section III-C, we notice that the error injection on different bits contributes differently to the output RRMSE eventually but the contribution proportion can be calculated based on (9) and (10). Similarly, we can also obtain the expected data disturbance of bit error on MSB. Then, we can calculate $RRMSE_{MSB}$ based on $RRMSE_{int8}$ with (19) and (20), and replace the standard random bit error injection with MSB-based error injection without compromising the analysis accuracy. Take VGGNet-11 and ResNet-18 quantized with int16 as examples. Fig. 10 shows the RRMSE obtained with both standard random bit error injection and MSB-based bit error injection. It confirms that the resulting RRMSE obtained with standard error injection and MSB-based error injection are linearly correlated as analyzed with the proposed statistical models. This approach can also be applied for straightforward model accuracy simulation. MSB-based fault simulation and standard fault simulation for both VGGNet-11 and ResNet-18 is presented in Fig. 11. It can be observed that the curves are quite similar and the difference is mainly induced by the scaled BER. In summary, given the same BER, MSB-based error injection can be scaled for standard bit error injection with negligible accuracy penalty while it reduces the total number of injected bit errors substantially and enhances the fault simulation speed accordingly

$$RRMSE_{MSB} = \sqrt{6}RRMSE_{int8}$$  \hspace{1cm} (19)  
$$RRMSE_{MSB} = \sqrt{12}RRMSE_{int16}.$$  \hspace{1cm} (20)

Third, according to the analysis in Section III-E, we notice that the influence of bit errors in different neurons and layers can be aggregated with (18) when we utilize RRMSE as the accuracy metric. Based on this feature, we can analyze the influence of complex error configurations with a disaggregated approach. For instance, we can analyze the influence of random bit errors in each layer independently and then construct the influence of random bit errors on multiple NN layers. We can also leverage the aggregation feature to scale RRMSE at lower BER to RRMSE at higher BER. In general, we can take advantage of this feature to obtain fault simulation of complex and large fault configurations with only a fraction of the fault configurations, which can greatly reduce the amount of fault simulation.

B. Fault Simulation Acceleration Examples

To quantize the fault simulation speedup, we take two typical fault simulation tasks as examples.

In the first task, we investigate how the model accuracy changes with the increase of BER. We still utilize VGGNet-11 quantized with int16 on ImageNet as the benchmark model. Suppose we want to explore the model accuracy when the BER
changes from 1E-7 to 1E-4 and we take 32 evenly distributed BER setups for the experiments. For each BER setup, we take 10,000 images from ImageNet to measure the model accuracy. With TensorFI fault injection framework [23], we need to conduct $3.2 \times 10^6$ inference with bit error injection. The fault simulation task can be accelerated with the first and the second approaches. It needs only four fault simulation points of RRMSE and model accuracy with $16 \times$ less bit error injection. The fault simulation time can be roughly reduced by $128 \times$. The results obtained with both a standard fault simulation and the accelerated fault simulation are shown in Fig. 12. They are quite close to each other, which confirms the quality of the accelerated fault simulation.

In the second task, we seek to find out the top-3 most fragile layers of an NN model such that an selectively hardening approach can be applied to protect the NN model with less protection overhead. We take VGGNet-11 quantized with int16 as a benchmark example and conduct fault injection at BER $= 5 \times 10^{-5}$. There are eight convolution layers in VGGNet-11. A standard fault simulation-based method needs to evaluate all the $C_8^3 = 56$ different combinations. For each configuration, we need to perform inference on around 10,000 images. Theoretically, we need to conduct $5.6 \times 10^5$ inference with random bit error injection. In contrast, with the second and the third acceleration approaches, we only need to conduct eight layerwise MSB-based error injection and perform inference on 1000 images for each error injection setup to obtain the NN RRMSE. Then, we can obtain RRMSE of all the different layer combinations immediately according to (18) in Section III. In this case, the fault simulation time can be reduced by $560 \times$. For NNs with more layers, this method can achieve exponential acceleration. In addition, we also evaluate the selected top-3 fragile layers based on the accelerated fault simulation approach. Suppose the top-3 fragile layers are fault-free, we can obtain the model accuracy with standard fault simulation and compare with that of all the 56 different combinations. The experiment result is shown in Fig. 13. It demonstrates that the selected top-3 layer are the most fragile layers of VGGNet-11, which is consistent with the results of standard fault simulation. In summary, the accelerated fault simulation not only reduces the execution time but also achieves high-quality results on this vulnerability analysis task.

VI. CONCLUSION

In this work, we observe that soft errors propagate across a large number of neurons and accumulate. With the observation, we propose to characterize the disturbance induced by the soft errors with a normal distribution model according to the central limit theorem and analyze the influence of soft errors on NNs with a series of statistical models. The models are further applied to analyze the influence of convolutional NN parameters on its resilience and verified with experiments comprehensively, which can guide the fault-tolerant NN design. In addition, we find that the models can also be utilized to predict many fault simulation results precisely and avoid lengthy fault simulation in practice. Given two typical fault simulation tasks, the model-accelerated fault simulation can be more than two orders of magnitude faster on average with negligible accuracy loss compared to standard fault simulation.

REFERENCES

[1] S. Pouyanfar et al., “A survey on deep learning: Algorithms, techniques, and applications,” ACM Comput. Surv., vol. 51, no. 5, pp. 1–36, 2018.
[2] K. Muhammad, A. Ullah, J. Lloret, J. Del Ser, and V. H. C. de Albuquerque, “Deep learning for safe autonomous driving: Current challenges and future directions,” IEEE Trans. Intell. Transp. Syst., vol. 22, no. 7, pp. 4316–4336, Jul. 2021.
[3] D. A. Hashimoto, G. Rosman, D. Rus, and O. R. Meireles, “Artificial intelligence in surgery: Promises and perils,” Ann. Surg., vol. 268, no. 1, pp. 70–76, 2018.
[4] S. O’Sullivan et al., “Legal, regulatory, and ethical frameworks for development of standards in artificial intelligence (AI) and autonomous robotic surgery,” Int. J. Med. Robot. Comput. Assist. Surg., vol. 15, no. 1, p. e1968, 2019.
[5] A. Reuther, P. Michaleas, M. Jones, V. Gadeppally, S. Samsi, and J. Kepner, “AI accelerator survey and trends,” in Proc. IEEE High Perform. Extreme Comput. Conf. (HPEC), 2021, pp. 1–9.
[6] A. Reuther, P. Michaleas, M. Jones, V. Gadeppally, S. Samsi, and J. Kepner, “Survey and benchmarking of machine learning accelerators,” in Proc. IEEE High Perform. Extreme Comput. Conf. (HPEC), 2019, pp. 1–9.
[7] A. Dixit and A. Wood, “The impact of new technology on soft error rates,” in Proc. Int. Rel. Phys. Symp., 2011, pp. 1–7.
[8] M. Shafique et al., “Robust machine learning systems: Challenges, current trends, perspectives, and the road ahead,” IEEE Design Test, vol. 37, no. 2, pp. 30–57, Apr. 2020.
[9] B. Reagen et al., “Ares: A framework for quantifying the resilience of deep neural networks,” in Proc. Des. Autom. Conf., 2018, pp. 1–6.
[10] G. Li et al., “Understanding error propagation in deep learning neural network (DNN) accelerators and applications,” in Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal. (SC), Denver, CO, USA, Nov. 2017, pp. 1–12. [Online]. Available: https://doi.org/10.1145/3126908.3126964
[11] D. Xu et al., “Persistent fault analysis of neural networks on FPGA-based acceleration system,” in Proc. IEEE 31st Int. Conf. Appl. Spec. Syst. Archit. Process. (ASAP), 2020, pp. 85–92.
D. Xu et al., “Reliability evaluation and analysis of FPGA-based neural network acceleration system,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 29, no. 3, pp. 472–484, Mar. 2021.

X. Xue, H. Huang, C. Liu, T. Luo, L. Zhang, and Y. Wang, “Winograd convolution: A perspective from fault tolerance,” in Proc. 59th ACM/IEEE Des. Autom. Conf., 2022, pp. 853–858.

C. Despres-Huitzil and B. Giraud, “Fault and error tolerance in neural networks: A review,” IEEE Access, vol. 5, pp. 17322–17341, 2017.

N. Humbatova, G. Jahangirova, G. Bavota, V. Riccio, A. Stocco, and P. Tonella, “Taxonomy of real faults in deep learning systems,” in Proc. ACM/IEEE 42nd Int. Conf. Softw. Eng., 2020, pp. 1110–1121.

S. Mittal, “A survey on modeling and improving reliability of DNN algorithms and accelerators,” J. Syst. Archit., vol. 104, Mar. 2020, pp. 1–19.

D. Xu et al., “Resilient neural network training for accelerators with computing errors,” in Proc. IEEE 30th Int. Conf. Appl. Spec. Syst., Archit. Process. (ASAP), 2019, pp. 99–102.

F. F. dos Santos, C. Lunardi, D. Oliveira, F. Libano, and P. Rech, “Reliability evaluation of mixed-precision architectures,” in Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA), 2019, pp. 238–249.

Y. He, P. Balaprakash, and Y. Li, “Fidelity: Efficient resilience analysis framework for deep learning accelerators,” in Proc. Ann. Int. Symp. Microarchit. (MICRO), Oct. 2020, pp. 270–281.

C. Liu, Z. Gao, S. Liu, X. Ning, H. Li, and X. Li, “Special session: Fault-tolerant deep learning: A hierarchical perspective,” in Proc. IEEE 40th VLSI Test Symp. (VTS), 2022, pp. 1–12.

D. Xu, F. Ahmed, and X. Oraiolglu, “SNR: Squeezing numerical range defuses efficiency vulnerability surface in deep neural networks,” ACM Trans. Embed. Comput. Syst., vol. 20, no. 5s, pp. 1–25, 2021. [Online]. Available: https://doi.org/10.1145/3477007

A. Mahmoud et al., “PyTorchFI: A runtime perturbation tool for DNNs,” in Proc. 50th Annu. IEEE/IFIP Int. Conf. Depend. Syst. Netw. (DSN-W), 2020, pp. 25–31.

Z. Chen, N. Narayanan, B. Fang, G. Li, K. Pattabiraman, and N. DeBardeleben, “TensorFI: A flexible fault injection framework for TensorFlow applications,” in Proc. Int. Conf. Softw. Rel. Eng. (ISSRE), 2020, pp. 426–435.

S. K. S. Hari, T. Tsai, M. Stephenson, S. W. Keckler, and J. Emer, “SASSIFI: An architecture-level fault-tolerant tool for GPU application resilience evaluation,” in Proc. IEEE Trans. Perform. Anal. Syst. Softw. (ISPASS), vol. 1, 2017, pp. 249–258.

D. Xu et al., “R2F: A remote retraining framework for AIoT processors with computing errors,” IEEE Trans. Very Large Scale Integrate. (VLSI) Syst., vol. 29, no. 11, pp. 1955–1966, Nov. 2021.

P. Pandey, P. Basu, K. Chakraborty, and S. Roy, “GreenTPU: Improving timing error resilience of a near-threshold tensor processing unit,” in Proc. 56th ACM/IEEE Des. Autom. Conf. (DAC), 2019, pp. 1–6.

B. Reagen et al., “Minerva: Enabling low-power, highly-accurate deep neural network accelerators,” in Proc. ACM/IEEE 43rd Annua. Int. Symp. Comput. Archit. (ISCA), 2016, pp. 267–278.

Z. Gao et al., “Soft error tolerant convolutional neural networks on FPGAs with ensemble learning,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 30, no. 3, pp. 291–302, Mar. 2022.

W. Li, X. Ning, G. Ge, X. Chen, Y. Wang, and H. Yang, “FIT-NAS: Discovering fault-tolerant neural architecture,” in Proc. 25th Asia South Pac. Des. Autom. Conf. (ASP-DAC), 2020, pp. 211–216.

X. Ning et al., “FIT-NAS: Discovering fault-tolerant convolutional neural architecture,” ACM Trans. Des. Autom. Electron. Syst., vol. 26, no. 6, pp. 1–24, 2021.

C. Liu et al., “HyCA: A hybrid computing architecture for fault-tolerant deep learning,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 41, no. 10, pp. 3400–3413, Oct. 2022.

J. J. Zhang, T. Gu, K. Basu, and S. Garg, “Analyzing and mitigating the impact of permanent faults on a systolic array based neural network accelerator,” in Proc. IEEE 36th VLSI Test Symp. (VTS), 2018, pp. 1–6.

Z. Chen, G. Li, K. Pattabiraman, and N. DeBardeleben, “BinFI: An efficient and reliable fault-injector for safety-critical computing systems,” in Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal., 2019, pp. 1–23. [Online]. Available: https://doi.org/10.1145/3295500.3356177

Y. Zheng, Z. Feng, Z. Hu, and K. Pei, “MindFI: A fault injection tool for reliability assessment of MindSpore applications,” in Proc. IEEE Int. Symp. Softw. Rel. Eng. Workshops (ISSREW), 2021, pp. 235–238.

B. Cheng and D. M. Titterington, “Neural networks: A Review from a statistical perspective,” Stat. Sci., vol. 9, no. 1, pp. 2–50, 1994.

H. White, “Learning in artificial neural networks: A statistical perspective,” Neural Comput., vol. 1, no. 4, pp. 425–464, Dec. 1989.

A. C.-C. Chang, R. H.-M. Huang, and C. H.-P. Wen, “CASSER: A closed-form analysis framework for statistical soft error rate,” IEEE Trans. Very Large Scale Integrate. (VLSI) Syst., vol. 21, no. 10, pp. 1837–1848, Oct. 2013.

H.-K. Peng, C. H.-P. Wen, and J. Bhadra, “On soft error rate analysis of scaled CMOS designs—A statistical perspective,” in Int. Conf. Comput.-Aided Des. Dig., 2009, pp. 1633–1637.

F. Libano et al., “Selective hardening for neural networks in FPGAs,” IEEE Trans. Nucl. Sci., vol. 66, no. 1, pp. 216–222, Jan. 2019.

G. Gambardella et al., “Efficient error-tolerant quantized neural network accelerators,” in Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst. (DFT), 2019, pp. 1–6.

S. S. Banerjee, J. Cyriac, S. Jha, Z. T. Kalbaczry, and R. K. Iyer, “Towards a Bayesian approach for assessing fault tolerance of deep neural networks,” in Proc. 49th Annua. IEEE/IFIP Int. Conf. Depend. Syst. Netw. Supplemental Vol. (DSN-S), 2019, pp. 25–26.

M. A. O. Ahmed, “Trained neural networks ensembles weight connections analysis,” in Proc. Int. Conf. Adv. Mach. Learn. Technologies Appl. (AMLTA), vol. 723, Cairo, Egypt, Feb. 2018, pp. 242–251. [Online]. Available: https://doi.org/10.1007/978-3-319-74690-6_24

A. Franchi, A. Bursuc, A. DeLuca, and I. Bloch, “TRADI: Tracking deep neural network weight distributions,” in Proc. 16th Eur. Conf. Comput. Vision (ECCV), Glasgow, U.K., Aug. 2020, pp. 105–121. [Online]. Available: https://doi.org/10.1007/978-3-030-58520-4_7

Haitong Huang received the B.S. degree in computer science from Xidian University, Xi’an, China, in 2021. He is currently pursuing the M.S. degree with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, and the Department of Computer Science, University of Chinese Academy of Sciences, Beijing.

His current research falls primarily in the field of fault-tolerant computing and neural network acceleration.

Xinghua Xue received the B.S. degree in electronic and information engineering from Shanxi University, Shaxi, China, in 2017, and the M.S. degree in IC engineering from Nankai University, Tianjin, China. She is currently pursuing the Ph.D. degree with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, and the Department of Computer Science, University of Chinese Academy of Sciences, Beijing.

Her research interests include fault-tolerant architecture design and robust deep learning.

Cheng Liu (Member, IEEE) received the B.Eng. and M.Eng. degrees from the Harbin Institute of Technology, Harbin, China, in 2007 and 2009, respectively, and the Ph.D. degree from the University of Hong Kong, Hong Kong, in 2016.

He is currently an Associate Professor with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, and the Department of Computer Science, University of Chinese Academy of Sciences, Beijing.

His research interests include FPGA-based reconfigurable computing, fault-tolerant computing, and custom computing.

Ying Wang (Member, IEEE) received the B.S. and M.S. degrees in electronic engineering from the Harbin Institute of Technology, Harbin, China, in 2007 and 2009, respectively, and the Ph.D. degree in computer science from the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), Beijing, China, in 2014.

He is currently an Associate Professor with ICT, CAS, and the Department of Computer Science, University of Chinese Academy of Sciences, Beijing.

His research interests include computer architecture and VLSI design, specifically energy-efficient architecture, and machine learning accelerators.
Tao Luo received the B.S. degree from the Harbin Institute of Technology, Harbin, China, in 2010, the M.S. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2013, and the Ph.D. degree from the School of Computer Science and Engineering, Nanyang Technological University, Singapore, in 2018.

He is currently a Research Scientist with the Institute of High Performance Computing, Agency for Science, Technology and Research, Singapore.

His current research interests include high-performance computing, green AI, quantum computing, hardware–software co-exploration, efficient AI, and its application.

Long Cheng received the B.E. degree from the Harbin Institute of Technology, Harbin, China, in 2007, the M.Sc. degree from the University of Duisburg–Essen, Duisburg, Germany, in 2010, and the Ph.D. degree from the National University of Ireland Maynooth, Kildare, Ireland, in 2014.

He is a Full Professor with the School of Control and Computer Engineering, North China Electric Power University, Beijing, China. He was an Assistant Professor with Dublin City University, Dublin, Ireland, and a Marie Curie Fellow with University College Dublin, Dublin. He has published more than 80 papers in journals and conferences, such as the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, IEEE Network Magazine, HPCA, and Euro-Par. His research focuses on distributed systems, deep learning, cloud computing, and process mining.

Prof. Cheng is a Co-Chair of the Journal of Cloud Computing.

Huawei Li (Senior Member, IEEE) received the B.S. degree in computer science from Xiangtan University, Xiangtan, China, in 1996, and the M.S. and Ph.D. degrees from the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), Beijing, China, in 1999 and 2001, respectively.

She has been a Professor with ICT, CAS since 2008. She is also a Professor with Peng Cheng Laboratory, Shenzhen, China. She has published over 180 technical papers and holds 27 Chinese Patents. Her current research interests include testing of VLSI/SOC circuits, design verification, and design for reliability.

Xiaowei Li (Senior Member, IEEE) received the B.Eng. and M.Eng. degrees in computer science from the Hefei University of Technology, Hefei, China, in 1985 and 1988, respectively, and the Ph.D. degree in computer science from the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), Beijing, China, in 1991.

He was an Associate Professor with the Department of Computer Science and Technology, Peking University, Beijing, from 1991 to 2000. In 2000, he joined ICT, CAS, as a Professor. He is also a Professor with the Department of Computer Science, University of Chinese Academy of Sciences, Beijing. He has coauthored over 280 papers in journals and international conferences and holds 60 patents and 30 software copyrights. His current research interests include VLSI testing, design verification, and dependable computing.