Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

An end-to-end automatic methodology to accelerate the accuracy evaluation of DNN under hardware transient faults*

Jiajia JIAO†‡, Ran WEN, Hong YANG

College of Information Engineering, Shanghai Maritime University, Shanghai 201306, China

†E-mail: jiaojiajia@shmtu.edu.cn

Received June 26, 2024; Revision accepted Sept. 18, 2024; Crosschecked

Abstract: Hardware transient faults are proven to have a significant impact on deep neural networks (DNNs), whose safety-critical-misclassification probabilities in autonomous vehicles, healthcare, and space applications are increased up to 4x. However, the inaccuracy evaluation using accurate fault injection is time-consuming and requires several hours and even a couple of days on a complete simulation platform. To accelerate the evaluation of hardware transient faults on DNNs, we design a unified and end-to-end automatic methodology, A-Mean, to take advantage of the silent data corruption (SDC) rates of basic operations, such as convolution, add, multiply, Relu, Maxpooling, etc., and a two-level mean mechanism to rapidly compute the overall SDC rate for estimating the general classification metric, accuracy and application-specific metric safety-critical-misclassification (SCM). More importantly, a max policy is used to determine the SDC boundary of non-sequential structures in DNNs. Then, the worst-case scheme is also used to further calculate the enlarged SCM and halved accuracy under transient faults via merging the static results of SDC with the original data from one-time dynamic fault-free execution. Furthermore, all of the steps mentioned above have been implemented automatically so that this easy-to-use automatic tool can be employed for the prompt evaluation of transient faults on diverse DNNs. Meanwhile, a novel metric fault sensitivity is defined to jointly characterize the variation of transient fault-induced higher SCM and lower accuracy. The comparative results with a state-of-the-art fault injection method on five DNN models and four datasets show that our proposed estimation method A-Mean achieves up to 922.80x speedup, with just 4.20% SCM loss and 0.77% accuracy loss on average.

Key words: Analytical model; Deep Neural Networks; Hardware transient faults; Fast evaluation; Automatic evaluation tool

https://doi.org/10.1631/FITEE.2400547 **CLC number:**

1 Introduction

Transient faults (Farjaminezhad et al. (2021a); BelÄĐeviÄĞ and StojanoviÄĞ (2022); Papadimitriou and Gizopoulos (2021)) primarily result from inherent circuit malfunctions (Jooshaki et al. (2023)) such as external radiation (Jiao et al. (2016); Al-haj Ahmad and Sedaghat (2022)) or internal electrical interference (Jung et al. (2022); Zhou et al. (2020)). These transient faults often occur randomly and cause high-performance loss or area/power overhead for detection and mitigation, so that some ignored faults have the potential to corrupt the program or lead to incorrect data. However, in safety-sensitive applications, such as autonomous vehicles (Jha et al. (2019)), healthcare (Adam et al. (2021)), and space applications (Li et al. (2021)), transient faults could

‡ Corresponding author

* Project supported by the Shanghai Pujiang Talent Program (No. 21PJD026)

lead to immeasurable loss of life and property. In recent years, deep neural networks (DNNs) (Laskar et al. (2022); Chen et al. (2020); Tan et al. (2023a)) have been adopted for safety-critical applications due to their remarkable problem-solving capabilities. However, these safety-critical applications necessitate dependable, robust, and efficient support from popular DNNs. Consequently, it is important to assess the accuracy of various DNN models under transient faults to guarantee their acceptable prediction quality (Dietrich et al. (2023); Ahmadilivani et al. (2023); Farjaminezhad et al. (2021b)).

Two categories of mainstream approaches are usually employed to evaluate the impact of faults on DNNs. **(1) Fault injection** involves injecting faults into DNNs intentionally many times so that the impacts can be quantified by the ratio of the number of injections with observed wrong results to the total number of injections, such as CAFI (Al-haj Ahmad and Sedaghat (2022)), Saca-FI (Tan et al. (2023b)), TensorFI (Chen et al. (2020)), and TensorFI+ (Laskar et al. (2022)). **(2) Fault free analysis** uses a few fault-free simulations and simple analytical models for fast evaluation of faultsâĂŹ impacts on DNNs, such as SERN (Ping et al. (2020)), APPRAISER (Taheri et al. (2023)), DeepVigor (Ahmadilivani et al. (2023)), Saca-AVF (Tan et al. (2023a)). The former is accurate but time-consuming, while the latter is fast but inaccurate. The DNN-driven safety-critical applications require not only high reliability for guaranteed safety but also high performance for real-time decisions. Therefore, fast fault-free analysis is preferred for safety-critical applications. However, how to improve the evaluation accuracy of the impacts of transient faults on DNNs with the existing advantage of fast speed is a challenge.

The advanced fault-free methods have used pure analytical models for fast evaluation. Ping et al. (2020) designed SERN to characterize the vulnerability characteristics of convolutional neural networks from the image information (data types, values, and the sign of data) and model structure (types of layers) quickly and accurately. Ahmadilivani et al. (2023) proposed DeepVigor to represent vulnerable and non-vulnerable ranges for each neuron to characterize the vulnerability factors for bits, neurons, and layers. However, two problems still require solving. One is that some dynamic information of the images input into diverse models is ignored, so the generality is limited. The other is that the specific safety-critical misclassification (SCM) of these safety-sensitive applications has not been considered in these analytical models.

To address these problems, we propose an **A**utomatic **M**ethodology for the fast **e**valuation of the **a**ccuracy of deep neural **n**etworks (A-Mean). This approach combines the dynamic information from one-time fault-free execution with static information from a fast analytical model for the accurate and fast evaluation of the general classification metric Accuracy and the application-specific metric SCM. Our main contributions are as follows:

(1) We design a unified two-level rapid and systematic assessment method, A-Mean, to evaluate the impact of transient faults on DNNs. Our approach can take advantage of one-time fault-free dynamic information, a static two-level analytical model, and a worst-case policy to effectively capture the inner-layer (data type, operator) and inter-layer (input feature map, depth, topology) fault impacts. It can achieve a high accuracy of up to 99.23% on average, with up to 922.80x speedup over the state-of-the-art fault injection.

(2) We conduct some experiments and compare our A-Mean with a single convolution SDC and no max-policy, respectively. As we implement three groups of SDC_{Conv} to replace other SDC_{op}, our results show that the different operators remain highly effective and consistently outperform the alternatives. The max-policy and no max-policy reveal distinct characteristics in topology, such as branches without hidden layers behaving like a sequential structure, while branches with hidden layers require consideration of both individual branches and their interactions. The results demonstrate that the max-policy clearly shows the impact among the topological branches.

(3) We develop an automated tool to accelerate the evaluation of Accuracy and SCM in DNNs under transient faults. This tool directly utilizes the modelâĂŹs printed structure and parameters, including operators and input feature maps, to automatically compute the SDC rate for each model. It is then combined with the Accuracy and SCM of a no-fault case to estimate the corresponding metrics under transient faults. The entire assessment process is completed in less than 0.12 seconds.

Meanwhile, a new metric, fault sensitivity, is also proposed for estimating transient fault impacts on the variation of Accuracy and SCM. We plan to make the artifact of A-Mean publicly available on https://github.com/breatrice321/A-Mean.

This paper is organized as follows. Section 1 provides a brief introduction. Section 2 describes the background and related work. The proposed method A-Mean is detailed in Section 3, while the comprehensive results and analysis are given in Section 4. The conclusion is summarized in Section 5.

2 Background and related work

2.1 Transient faults impact on DNNs

Transient faults, also known as soft errors, are caused by high-energy neutron or alpha particle strikes in integrated circuits (Mukherjee (2008)). Besides system execution interruption or corruption, these transient fault-induced bit upsets may also silently corrupt data and lead to erroneous computation results, known as silent data corruption. DNNs are frequently used in safety-critical applications and are significantly affected by transient faults.

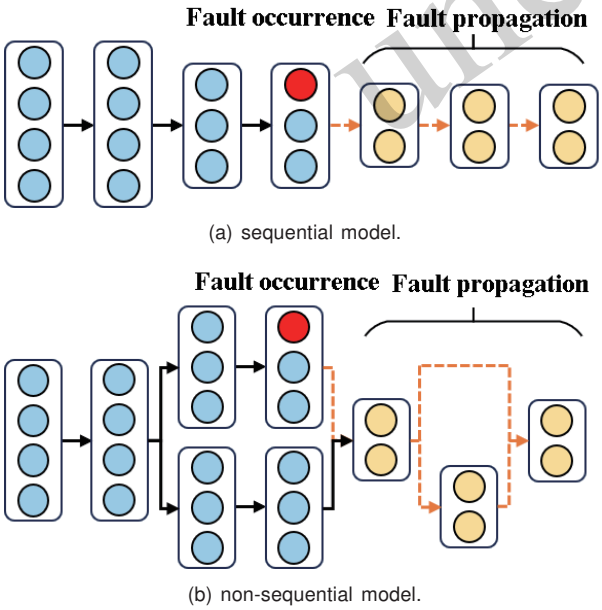


Fig. 1 Fault occurrence and propagation in DNNs.

DNNs are composed of neural units in a multi-layer structure, where the output features from one hidden layer serve as input features for the subsequent layer, enabling the network to progressively capture features, shown in Fig. 1. Through a series of feature transformations across multiple layers, DNNs can map the inherent characteristics of input samples to a distinct feature space. Consequently, the representation of input features can be enhanced well. However, due to the multi-layer hierarchical architecture (Shi et al. (2023); Liang et al. (2023)), such as in the sequential models presented in Fig. 1a and the non-sequential models depicted in Fig. 1b, the hardware transient faults in a red node of an arbitrary hidden layer can spread through the network, potentially affecting the yellow nodes of successive layers and the final output (Laskar et al. (2022)).

This distributed and parallel architecture (Ruospo et al. (2022)), as well as inherent redundancy (Ruospo et al. (2023)) from over-provisioning, also gives DNNs some fault tolerance (Sun et al. (2021); Jung et al. (2022); Camponogara Viera et al. (2017)). However, the inherent fault tolerance of DNNs is relatively limited under more hardware transient faults (Ruospo et al. (2022, 2023)). When injecting faults into the neural units, the resulting errors propagate in the model and influence the final prediction accuracy and SCM. For example, VGG16 achieves 70.43% accuracy and 2.11% SCM under no faults, while transient faults can lead to 68.03% accuracy and 3.06% SCM. Therefore, we can observe that DNNs exhibit a degree of resilience against faults, and the majority of faults are masked, even up to 96% (Laskar et al. (2022)). However, 36.25% of the changed data affected by fault injection are critical to safety. Safety-critical applications (e.g., autonomous vehicles, healthcare, and space) cannot tolerate any safety-critical misclassifications, as these faults can result in catastrophic consequences. Hence, it is essential to accurately evaluate the impact of faults on these applications.

2.2 Faults models

Fault time. In this work, faults are assumed to occur randomly in DNNs during the fault-sensitive inference phase. Compared with the one-time training phase, the inference phase is not easily verified by the results of the trained models. More importantly, this is aligned with prior work (Laskar et al. (2022); Chen et al. (2020)).

Fault location. Transient faults probably occur in the input, parameters, and functions in a random hidden layer of DNNs. Therefore, we use this

configuration with a diversity of fault locations. It is noted that to track the propagation of faults quickly, faults are often injected directly into the output values of a hidden layer in DNNs (Laskar et al. (2022); Chen et al. (2020)).

Fault type. The increasing multi-bit upset has only an 8% probability of causing SDC (Sangchoolie et al. (2017)); the work of Sangchoolie et al. (2017) shows that multi-bit upset-induced errors exhibit similar error propagation patterns and silent data corruption probabilities to single-bit upset-induced errors. Therefore, this work adopts the often-used single-bit upset as done by Laskar et al. (2022).

2.3 Evaluation approaches of hardware transient faults on DNNs

Hardware transient faults make the reliable design of computer architecture increasingly significant (Du (2022); Venkatesha and Parthasarathi (2022); Raj et al. (2020); Eeckhout (2022); Yao et al. (2022)). To achieve cost-effective protection, it is necessary to evaluate fault impacts quickly and accurately using two categories of methods: fault injection and fault-free analysis.

Fault injection requires a number of dynamic executions to calculate the ratio of the number of executions with incorrect results to the total number of executions for accurate estimation. Laskar et al. (2022) investigated the impact of DNN misclassification caused by hardware transient faults in safety-critical applications and extended a fault injector for the TensorFlow application to support fault injections on DNN models using TensorFI+. Gavarini et al. (2023) propose a smart, accurate, and unintrusive fault-injector SCI-FI to reduce the evaluation procedure using fault-dropping and delayed start. Zheng et al. (2021) presented a tool called MindFI to cover a variety of faults in machine learning programs written in Mindspore. These accurate fault injection methods involve a long time, performing many experiments, and are limited to specific hardware architectures. Thus, they find it difficult to satisfy the real-time requirements and good generality of diverse safety-critical applications.

A fast fault-free analysis is preferred for evaluating transient impacts on diverse DNNs for its high speed and good generalization. This kind of evaluation method primarily exploits the deep insights of transient fault propagation in the underlying system by using a few formulas for fast analytical models. Ping et al. (2020) proposed an analytical model SERN to assess soft error impacts on CNNs using only a small number of CNN parameters. Ahmadilivani et al. (2023) presented a fine-grain, metric-oriented evaluation method, DeepVigor, that represents vulnerable and non-vulnerable ranges for each neuron to characterize the vulnerability factors for layers. Tan et al. (2023a) presented Saca-AVF to conduct a quantitative analysis of a CNN acceleratorâĂŹs reliability from an architectural perspective. However, the following two problems still exist: (1) The dynamic information of fault propagation through the input image into diverse DNN models is ignored so that the generality of pure analytical models for new DNN models is limited; (2) SCM, as an application-specific metric for these safety-sensitive applications, has not been considered in these analytical model-based evaluations.

To address these challenges, an analytical model-driven method, A-Mean, is proposed for rapidly and accurately assessing the transient faultsâĂŹ impacts on DNN inaccuracy. The proposed approach uses one-time fault-free execution to capture the basic no-fault SCM information and then combines it with a two-level analytical model for SCM evaluation under transient faults. More importantly, this well-generalized work is implemented to serve as an end-to-end automatic tool for fast and accurate evaluation.

2.4 Reliability metrics: SDC rate and SCM

SDC is an intricate challenge in computer systems and manifests as errors occurring across various stages of data storage and transmission (Chen et al. (2020); Ramzanpour and Ludwig (2020); Papadimitriou et al. (2023)). The transient fault-induced bit upset remains undetectable at the time of occurrence and is only unveiled during subsequent data access, posing significant concerns. The primary sources of SDC include hardware faults and software bugs. This paper focuses on the hardware transient fault-induced SDC. SDC rate is the probability of output of an SDC result under a random fault occurrence. It can be calculated as the ratio of the number to the total number of executions using fault injection. For example, the total number of fault injections into convolution operations is 1000, and the number of SDC results is 220. Its corresponding SDC rate is

denoted SDC_{conv} and is equal to 0.22.

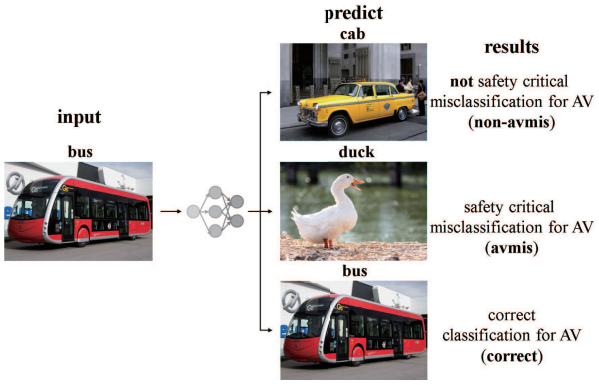


Fig. 2 Definition of non-avmis, avmis and correct classification.

SCM is a specific metric for measuring the misclassifications of safety critical data in DNNs, and we redefine the meaning of SCM based on Laskar et al. (2022). Unlike the general metric Accuracy, it emphasizes the safety-related impacts and can be calculated by the ratio of the number of safety-aware misclassifications to the total number of classifications. As shown in Fig. 2, if a bus is mistakenly identified as a cab, this would be considered a misclassification. However, in DNN classification of autonomous vehicles, whether it is a bus or a cab would not significantly impact safety (non-avmis). On the other hand, if a bus is misclassified as a duck, the brake that was supposed to be applied will be cancelled, which will cause significant safety hazards (avmis). Therefore, A-Mean is proposed for evaluating this metric on diverse DNNs. In order to explain the SCM calculation procedure, we give a simple example. It is assumed that there are 100 tested data results, among which 70 are correctly classified (correct) and 30 are incorrectly classified. Among the 30 misclassifications, three are misclassifications that are very important for security (avmis), and 27 are misclassifications that are not important for security (non-avmis). Therefore, Accuracy=70/100=0.7, abbreviated as Acc, is shown in eq(1), and InAcc=1-Acc. SCM=3/100=0.03 as shown in eq(2), and nonSCM=1-SCM.

$$Acc = \frac{correct}{avmis + non - avmis + correct} \quad (1)$$

$$SCM = \frac{avmis}{avmis + non - avmis + correct} \quad (2)$$

3 Proposed analytical model A-Mean

3.1 Overall framework

Our proposed A-Mean includes three parts, as Fig. 3 shows. It makes good use of both the dynamic execution results and static analysis information for fast, accurate, and fully automatic evaluation.

(1) One dynamic execution. This part involves the basic Acc and SCM of DNNs without faults. Transient faults enlarge InAcc because the original classification results of Acc are partially changed into InAcc by faults. Furthermore, these transient faults also worsen the SCM of the classification. Therefore, it is necessary to obtain the basic Acc and SCM values. This procedure only requires one dynamic execution to obtain the classification results.

(2) Static two-level mean calculation. After the one execution calculates the basic Acc and SCM under no faults, the next step is to compute the expansion rate under transient faults. For speed and simplicity, we consider the overall SDC rate of the specific DNN structure as the expansion rate to make partial Acc into InAcc, and nonSCM into SCM. The two-level mean calculation includes the inner-level and inter-level SDC stages shown in Fig. 3. If a fault occurs in a layer of the DNN, the final output result is influenced by various factors such as operations, input feature maps, the depth of networks, and topologies. Therefore, this paper considers these factors in a joint way. As for the different kinds of operations in each layer, we develop the first-level mean calculator to consider the operator type and frequency. Then, the second-level mean calculation uses the normalized input size combined with depth as the inter-layer weight and the different SDC layer computation policies for characterizing different DNN topologies. Further details are given in Section 3.2.

(3) Fusion using worst-case policy. Based on the Acc and SCM from the one dynamic execution, as well as the overall SDC obtained from the two-level mean calculation, the worst-case policy is used to merge these two kinds of information in eq(3) and eq(4). Acc_{no_fault} represents the accuracy without faults, and $InAcc_{no\_fault}$ represents the inaccuracy without faults. The sum of the two values is equal to 1. $nonSCM_{no\_fault}$ represents the nonSCM value without faults while SCM_{no_fault} is the SCM

value without faults. The sum of $nonSCM_{no\_fault}$ and SCM_{no_fault} is 1.

In the worst case, transient faults do not alter the misclassification results of $InAcc_{no fault}$, nor do they affect the SCM_{no_fault}. Meanwhile, some of the Acc_{no_fault} becomes $InAcc_{no\_fault}$ using the overall SDC as the expansion rate in eq(3).

However, compared with the influence of SDC on Acc_{no_fault}, $nonSCM_{no\_fault}$ needs to be considered in two respects. As illustrated in eq(1) and eq(2), SCM focuses exclusively on avmis, while Acc only considers correct classifications, which results in non-avmis being overlooked. Therefore, in eq(4), non-avmis and correct classifications are considered separately.

For the non-avmis category, which represents misclassifications that do not significantly impact safety, we assume that after fault injection, there is a $1/2$ probability of it turning into avmis and a $1/2$ probability of it turning into another non-avmis.

For the correct category, we draw an analogy to patients with preexisting conditions having a higher mortality rate compared to healthy individuals. Thus, the probability of a correct category turning into avmis is lower than that of a non-avmis category turning into avmis under faults. Specifically, we assume that $1/(2+\varepsilon)$ ($\varepsilon>0$) of correct cases will turn into avmis, with the remaining proportion turning into the non-avmis category.

$$\begin{aligned} Acc_{with\_fault} =& 1-\big(Acc_{no\_fault}\times SDC \\ &+Inacc_{no\_fault}\big) \end{aligned} \tag{3}$$

$$\begin{aligned} SCM_{with\_fault} =& \big[\big(nonSCM_{no\_fault} \\ &- Acc_{no\_fault}\ \big)\times\frac{1}{2} \\ &+ Acc_{no\_fault}\times\frac{1}{2+\varepsilon}\ \big]\times SDC \\ &+ SCM_{no\_fault} \end{aligned} \tag{4}$$

3.2 Details in two-level mean module

Our estimation approach, A-Mean, consists of two levels of computing as depicted in Fig. 3, as follows: (1) the first-level inner-layer mean calculation for each layer SDC_{layer} ; and (2) the second-level inter-layer mean calculation for the overall SDC.

(1) First-level mean calculation for each layer. It considers the high frequently used operations SDC_{op} as the fundamental infrastructure and the operations frequency as their corresponding inner-layer weight $Weight_{op}$ to compute SDC_{layer} using eq(5).

$$SDC_{layer\_i}=\sum_{j=0}^{j=k-1} Weight_{op_j}\times SDC_{op_j} \tag{5}$$

As for k operations in the $i-th$ layer of DNN, the mean value of all the k basic SDC_{op_j} by its operation frequency $Weight_{op}$. The weight of the $j-th$ operation $Weight_{op_j}$ is the ratio of the number of $j-th$ operations to the total number of k operations in eq(6). It is noted that our estimation method A-Mean assumes the equal fault occurrence probability of different operations such as convolution (conv) and Rectified Linear Unit (ReLu). The physical implementation determines their area consumption and fault occurrence probability. It can be configured to be suitable for different implementations by changing the weight easily.

$$\begin{cases} Weight_{op_j}=\frac{N_{op_j}}{\sum_{j=0}^{j=k-1}N_{op_j}} \\ while\ \sum_{j=0}^{j=k-1}Weight_{op_j}=1 \end{cases} \tag{6}$$

In this paper, the SDC_{op} for each type of operator is shown in Table 1 from our fast estimation method, described in Section 3.3. These SDC_{op} values also can be estimated using other available evaluation methods.

Table 1 the SDC_{op} of different operators

Operations (OP)	A-Mean	Chen et al. (2020)
Conv	0.25	0.22
Add	0.125	0.1
Sub	0.125	0.02
Mul	0.125	0.15
Div	0.125	0.15
ReLu	0.375	0.3
Max Pooling	0.25	0.35
Average Pooling	1	1

(2) Second-level mean calculation for joint layers. The output of the first-level mean value for each layer SDC_{layer} is the input to the second level to characterize the correlation between different layers in eq(7).

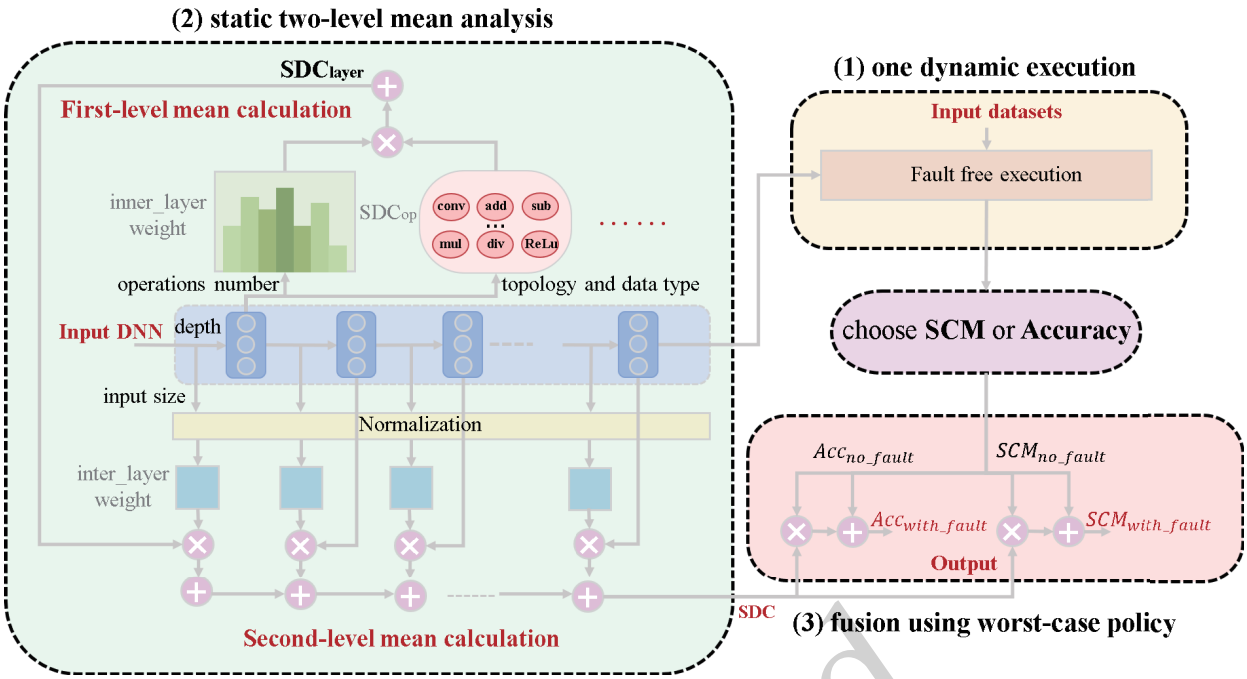


Fig. 3 Comparison of different methods in terms of outlier detection accuracy

$$SDC = \sum_{i=0}^{i=n-1} Weight_{layer_i} \times SDC_{layer_i} \quad (7)$$

Our proposed A-Mean uses the input feature map (W, H, C) of each layer as its inter-layer weight for accurate evaluation, where W represents the input width, H represents the input height, and C represents the number of input channels. Due to the varying inputs and possible data fluctuations, we normalize the inter-layer weights and calculate the mean value of all n layers in eq(8).

$$\begin{cases} Weight_{op_j} = \frac{W_i \times H_i \times C_i}{\sum_{j=0}^{j=n-1} W_i \times H_i \times C_i} \\ while \ \sum_{i=0}^{i=n-1} Weight_{layer_i} = 1 \end{cases} \quad (8)$$

The depth of the network is taken into account in A-Mean. From Li et al. (2017), we found that the impact of SDC on DNNs is influenced by the layer's position: layers that are shallower (closer to the input) are more affected. This is because faults occurring in earlier layers have a higher probability of propagating through to the subsequent layers. In our work, we selected inter-layer weights and used the input feature map to amplify the influence of the earlier layers. However, the significant differences in the input feature maps between the earlier and later layers of deep networks lead to substantial discrepancies in inter-layer weights. For instance, in VGG16, these differences can be larger by as much as 785 times. To address this issue, we employ an inverse depth weighting strategy, as illustrated in eq(10). Additionally, as indicated in eq(9), given the varying depths of models like VGG16 and ResNet50, we aim to prevent excessively deep networks from resulting in very small weights. Therefore, the influence of depth is adjusted by using $log(depth_{layer_i}) + \varepsilon'$. This approach helps to moderate the effect of depth while accounting for differences across layers.

$$inf_{layer_i} = depth_{layer_i} - log(depth_{layer_i}) + \varepsilon' \quad (9)$$

$$Weight_{layer_i} = \frac{Weight_{layer_i}}{inf_{layer_i}} \quad (10)$$

3.3 SDC_{op} for each operation

As the fundamental parameter for SCM estimation in A-Mean, SDC_{op} estimation and configuration are significant. Furthermore, SDC_{op} is the key factor affecting the final SDC as we compute final SDC with different SDC_{op}. For example, as shown in Table 1, the first line represents the SDC calculated using our method, detailed later in this paper.

In contrast, the second line corresponds to the data from Chen et al. (2020), which was derived from fault injection experiments measuring the SDC for each operation. Based on two different SDC_{op}, we calculated the SDC rate for DNNs, such as VGG16, to be 0.0205 and 0.0182, respectively. Therefore, suitable SDC_{op} are supposed to be provided according to model operations and other influence factors.

Previous works (Li et al. (2017); Zheng et al. (2021)) have proven that data type has a significant impact on SDC_{op}. This paper uses float type. Floating-point numbers comprise three parts: sign bit, exponent bit, and tail bit. The SDC_{op} generated by floating-point numbers with different bits will yield different results. However, they have a common point that the error of SDC is mostly caused by the high-order part of the exponent bit in the floating-point number (Li et al. (2017); Zheng et al. (2021)). Therefore, the SDC_{basic} is defined as the ratio of the higher-order bits of the exponent to the total number of bits as shown in eq(11).

$$SDC_{basic} = \frac{high-order\ exponent}{sign + exponent + tail} \tag{11}$$

As for float32, the exponential part is 8 bits. The highest 4 bits of the exponential part are assumed to influence SDC significantly. Therefore, the basic value of SDC_{basic} is set as 0.125=4/32=1/8. Due to the selection of multiple operators in this article, we categorize them into four types and explain the policies for SDC calculation of various operations.

(1) Common operations. Addition (Add), subtraction (Sub), multiplication (Mul), and division (Div) are common operations in DNNs. Addition is often used to connect data, such as residual connections; multiplication is often used in weight updates, activation functions, and other aspects; and subtraction and Division usually appear in data normalization. These simple operations are fundamental for more complex operations. In this paper, the SDC_{Add}, SDC_{Sub}, SDC_{Mul}, and SDC_{Div} are set to SDC_{basic} as mentioned above.

(2) Convolution operation. The convolution (Conv) operation is mainly used for convolution calculation, and the essence of convolution calculation is multiplication and addition. Therefore, we take the sum of SDC_{Add} and SDC_{Mul} as the convolution SDC, so the SDC_{Conv}=SDC_{basic}ÃŮ2.

(3) ReLu operation. One of the most commonly used activation functions is ReLu, and its expression can be found in eq(12):

$$ReLu = max(0, w^T x + b) \tag{12}$$

ReLu mainly consists of two parts; one is 0 (x<0), and the other is $w^T x + b$ (x>0). When x<0, if a bit upset occurs and 0 becomes another value, then all results in this part will become incorrect. Thus, we set SDC_{ReLu_part}=0.5 as x<0. When x>0, this part of the operation consists of two parts—multiplication and addition—and this part SDC_{ReLu_part}=SDC_{basic}ÃŮ2. Therefore, SDC_{ReLu} is set to eq(14), which uses the mean value of two parts to fully consider the impact of the bit upset.

$$SDC_{ReLu\_part} = \begin{cases} SDC_{basic} \times 2 & x > 0 \\ 0.5 & x < 0 \end{cases} \tag{13}$$

$$SDC_{ReLu} = \frac{SDC_{basic} \times 2 + 0.5}{2} \tag{14}$$

(4) Max pooling and Average pooling. Max pooling and Average pooling are two common pooling operations. The difference is that Average pooling is mostly placed at the end of the result output and is used to obtain the average value of the data in the current filter. If this part of the data undergoes bit upset, it will affect the final result directly. Therefore, SDC_{avg_pool}=1. On the other hand, Max pooling often appears together with convolution layers and is used to obtain the maximum value of the data in the filter. Meanwhile, Max pooling has a similar function to convolution layers, which are used to extract features. Here, SDC_{max_pool}=SDC_{Conv}.

3.4 Max-policy for non-sequential DNN topologies

The topology of DNNs is not always in a simple sequential structure, as Fig. 1a depicts. Therefore, it is necessary to consider different branches in non-sequential DNNs, as illustrated in Fig. 4. The standard two-level mean calculation for sequential structure is given in Fig. 4a. Considering the branch distribution of non-sequential DNN topologies, we further categorize the non-sequential structures into

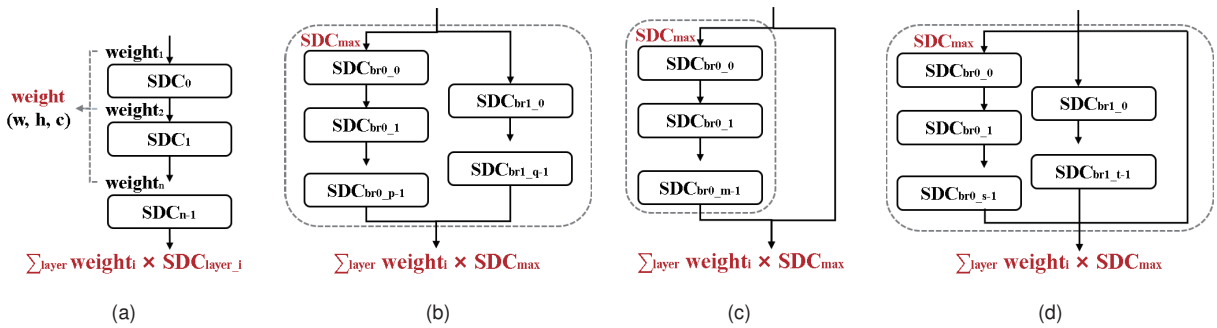


Fig. 4 Various DNN topologies. (a) Sequential structure, (b) Branch topology 1: Both branches have hidden layers, (c) Branch topology 2: One branch contains hidden layers, (d) Branch Topology 3: Some branches have hidden layers and some do not.

two types, as depicted in Fig. 4b and Fig. 4c. Interestingly, Fig. 4d can be considered a combination of Fig. 4b and Fig. 4c for more branches in non-sequential structures.

A max-policy is designed to handle these branches in Fig. 4b and Fig. 4c for non-sequential DNN models. To guarantee the reliability, we compute the maximum of multiple SDC_{layer} for each branch structure and estimate the upper bound to capture the sequential and parallel modes jointly. The worst-case scenario for fault propagation and the interdependencies between the branches motivate the use of the max-policy. As Fig. 4b shows, the maximum value of $\{SDC_{br0\_0}, SDC_{br0\_1}, \ldots, SDC_{br0\_m-1}\}$ is set to SDC_{br}. In a similar way, in SDC_{br} of Fig. 4c, the maximum SDC_{layer} of $(p+q)$ layers in two branches. After the parallel computation of different branches, the maximum SDC_{br} replaces all the SDC_{layer} in the branch for the overall SDC calculation and the final estimation.

More complex combinations in the diverse DNNs can be composed of three basic topologies in Figs. 4a-4c and estimated by the two-level mean mechanism for sequential structures and max-policy for branches in non-sequential structures. Therefore, the proposed A-Mean considers the different kinds of topologiesâĂŹ impacts on the fault propagation in DNNs for accurate Acc_{with_fault} and SCM_{with_fault} estimation.

3.5 End-to-end automatic tool

To use our A-Mean for fast and accurate evaluation conveniently, we developed the end-to-end automatic tool, as shown in Fig. 5.

As depicted in Fig. 5, the essential information is extracted from the DNN, such as the input feature map, depth, operations, and topology, to calculate the two-level mean weights described in Section 3.2. However, the information varies for different models, making manual processing complex and time-consuming. Therefore, we implement an end-to-end automatic tool to assist in preprocessing data, aiding in the extraction and organization of diverse information from the various networks mentioned above. The final Acc_{with_fault} and SCM_{with_fault} is obtained from the one-time dynamic execution and the SDC from the two-level mean calculation. Our end-to-end automatic tool completes the entire calculation process in less than 0.12 seconds. It is highly efficient and suitable for real-time requirements of diverse safety-critical applications.

3.6 Validating A-Mean by fault injection

To verify the effectiveness of the proposed A-Mean, we conduct the comparative experiments using state-of-the-art (SOTA) fault injection TensorFI+ (Laskar et al. (2022)) and fault-free analysis A-Mean on five DNN models. The steps for validating A-Mean are as follows:

(1) Random fault injection for TensorFI+. TensorFI+ randomly injects single-bit upset to a randomly selected hidden layer of DNNs during the inference phase.

(2) Validating A-Mean. We define three metrics from three perspectives (Speed, Accuracy, and SCM) to validate A-Mean, which are the speedup of estimation time, SCM loss, Accuracy loss, and fault sensitivity.

The speedup of estimation time reflects the acceleration of our automatic evaluation method over

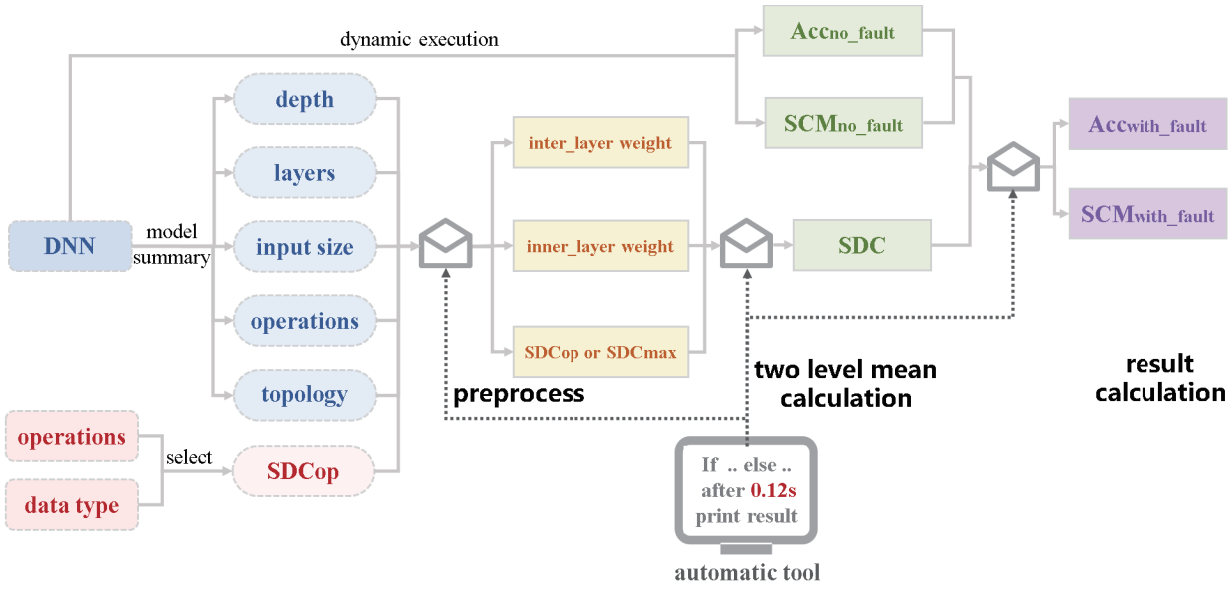


Fig. 5 End-to-end automatic tool.

the SOTA fault injection method, and this paper uses speedup, calculated by eq(15), to measure it. $Runtime_{AMean}$ is the total runtime of A-Mean, including the time of one-time dynamic execution and the estimation time, while $Runtime_{TensorFI+}$ represents the total time of multiple fault injections using TensorFI+.

$$Speedup = \frac{Runtime_{TensorFI+}}{Runtime_{AMean}} \tag{15}$$

Acc_{AMean} and SCM_{AMean} is estimated by the proposed A-Mean method using one-time dynamic execution, while $Acc_{groundtruth}$ and $SCM_{groundtruth}$ represent the results from N random fault injections. ÎŤAcc and ÎŤSCM are calculated to give the absolute deviation between ground truth and A-Mean. Accuracy loss is one of the metrics for measuring the effectiveness of various estimation methods in eq(16), and SCM loss is another metric for evaluating the model in eq(17).

A new metric called fault sensitivity is also introduced to evaluate the impact of fault injection on safety-critical aspects of the model. To quantify the magnitude of changes in the data that are critical to safety, we measure the ratio of SCM variation to Accuracy variation without faults and with faults, as described in eq(18). This approach helps us to observe how faults affect safety-critical aspects of the model.

$$\begin{cases} \Delta Acc = \left| Acc_{A\_Mean} - Acc_{groundtruth} \right| \\ Accuracy\ loss = \frac{\Delta Acc}{Acc_{groundtruth}} \end{cases} \tag{16}$$

$$\begin{cases} \Delta SCM = \left| SCM_{A\_Mean} - SCM_{groundtruth} \right| \\ SCM\ loss = \frac{\Delta SCM}{SCM_{groundtruth}} \end{cases} \tag{17}$$

$$\begin{cases} \Delta SCM' = \left| SCM_{with\_fault} - SCM_{no\_fault} \right| \\ \Delta Acc' = \left| Acc_{with\_fault} - Acc_{no\_fault} \right| \\ Sensitivity = \frac{\Delta SCM'}{\Delta Acc'} \end{cases} \tag{18}$$

4 Results and analysis

4.1 Experiment setup and configuration

The experimental configuration is shown as follows: all the experiments in this paper run on our private server with Ubuntu Linux 18.04.6 LTS on an Intel XeonÂő Silver 4210 2.2GHz processor with 125.5 GB of DDR4 memory. The datasets include ImageNet, CIFAR100, CIFAR10 with 10000 images, and STL10 with 8000 images.

In order to validate the effectiveness of A-Mean through fault injection, all the data from four datasets are utilized to calculate the metrics. This reproduced work is different from the study by Laskar

et al. (2022), which randomly selected a subset of data.

4.2 Estimation speed comparison using runtime and speedup

The runtime of the proposed A-Mean is relatively shorter than TensorFI+, proposed by Laskar et al. (2022), as shown in Fig. 6. Fault injection-based TensorFI+ consumes up to 98 hours, while A-Mean achieves a one to two orders of magnitude improvement over TensorFI+, up to 23.18 922.80x speedup across four different datasets. Due to only one dynamic execution in A-Mean instead of multiple executions in FI, the runtime is reduced dramatically to around 0.11 0.5 hour, while the static estimation of the two-level mean calculation and worst-case policy, as well as the pre-process of network for static estimation, are less than 0.12 seconds and negligible. Therefore, our proposed A-Mean is more suitable for the fast estimation of transient faultsâĂŹ impact on DNN inaccuracy.

Furthermore, A-Mean estimates the Acc_{with_fault} and SCM_{with_fault} of non-sequential networks, such as MobileNetV2 and ResNet50, and achieves higher speedup, 155.28x and 189.42x, 540.28x and 425.83x, 323.29x and 310.65x, and 828.24x and 922.80x in ImageNet, CIFAR100, CIFAR10, and STL10, respectively. Instead, if the test is on the sequential structures of VGG16, VGG19, and MobileNet, A-Mean can obtain lower acceleration. This phenomenon results from the fact that the more layers and complex structures in non-sequential DNNs require more, longer, time-consuming fault injections than simple sequential structures. Therefore, the proposed A-Mean performs better in terms of speed advantage in the complex DNN models.

4.3 Estimation accuracy using SCM loss & Accuracy loss

The estimation results of Acc_{with_fault} and SCM_{with_fault} by our proposed A-Mean and SOTA fault injection TensorFI+ (Laskar et al. (2022)) are shown in Fig. 7. TensorFI+ provides $Acc_{groundtruth}$ and $SCM_{groundtruth}$ (this paper reproduces the fault injection work to align the experiments) while our A-Mean computes the estimated Acc_{A-Mean} and SCM_{A-Mean}.

The Accuracy estimated by our proposed A-Mean is very close to the ground truth from fault injection. As shown in Fig. 7a, Accuracy loss in eq(16) is very low, from 0.02% to 1.42% on four varying datasets.

Similarly, SCM loss is also very low, and A-Mean can provide accurate estimation as fault injection. As shown in Fig. 7b, the SCM loss values range from 0. 42% to 9.85%, from 0.09% to 7.80%, from 2.96% to 14.21%, and from 0.46% to 2.13%, respectively, on the five DNNs across four different datasets.

Consequently, our A-Mean method can effectively accelerate the transient faults assessment in DNNs with guaranteed accuracy. The primary reasons lie in two points. One is the fusion of dynamic execution and fast static analysis. The other is that the two-level mean mechanism considers the input feature map, depth, operators, and topologies of DNNs very well to characterize the fault impacts on the final SCM.

4.4 Fault Sensitivity of avmis under various DNN models and datasets

A novel metric fault sensitivity is introduced to measure the ratio of $\Delta SCM^{'}$ to $\Delta Acc^{'}$ without fault and with faults. It can directly reflect the fault impacts on increasing SCM over decreasing Accuracy.

As illustrated in Table 2, the fault sensitivity results of fault injection (ground truth) is compared with those using A-Mean. The fault sensitivity estimated by A-Mean across four datasets are 55.80% 59.49%, 62.62% 122.45%, 43.63% 47.80%, and 78.34% 98.11%, respectively, while the ground truth sensitivity values are 36.25% 42.31%, 38.10% 55.00%, 30.00% 45.63%, and 0.00% 25.00%. These comparisons reveal variations in assessment outcomes to some extent. For example, the ImageNet (first group) demonstrates relatively high evaluation results, while the CIFAR100 (second group) shows anomalous data with a value of 1.22. Meanwhile, the CIFAR10 (third group) yields more accurate fault sensitivity results. However, the STL10 (forth group) presents the complementary fault sensitivity between ground truth and A-Mean.

One primary reason for these discrepancies lies in the different classification accuracy levels of the datasets, as highlighted in Table 2. CIFAR10 exhibits substantially higher accuracy compared to the other three datasets, resulting in more precise sensi-

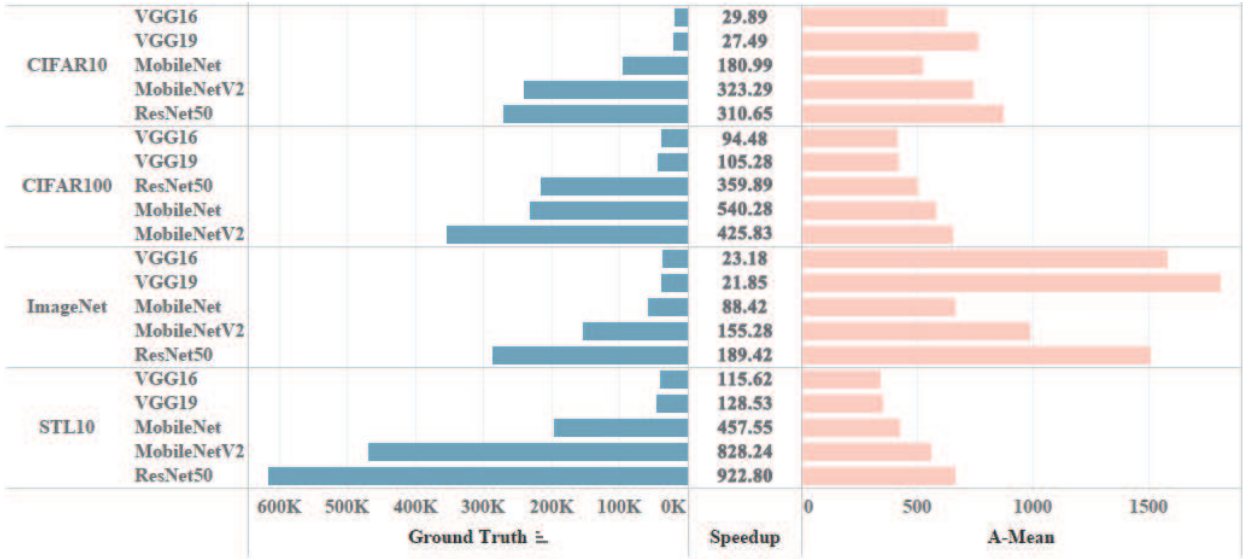
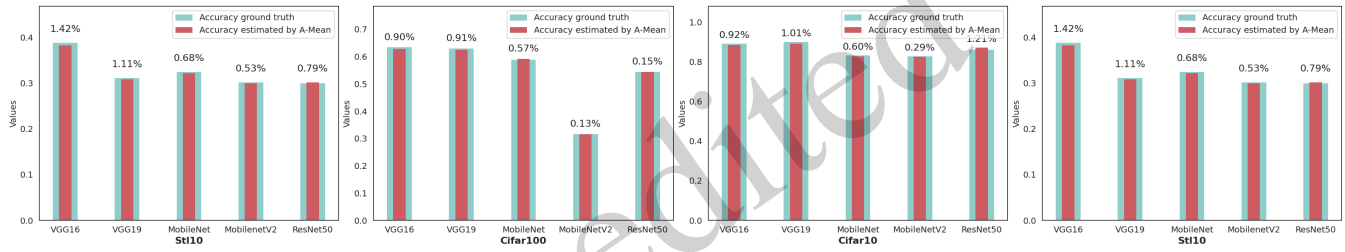
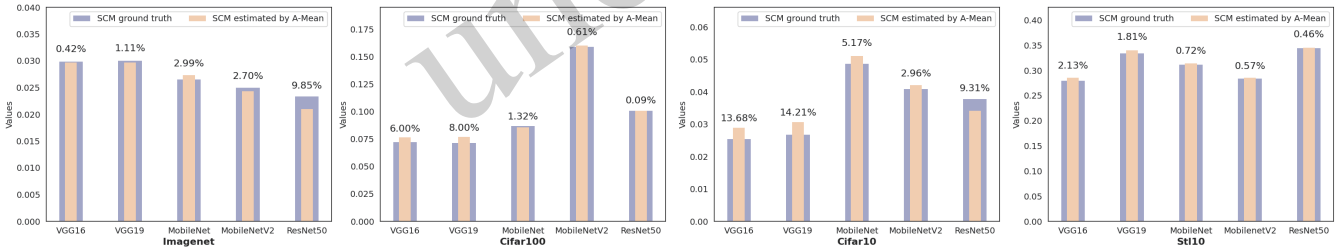


Fig. 6 Speedup between A-Mean and TensorFI+ (ground truth).



(a) Accuracy loss on four datasets



(b) SCM loss on four datasets.

Fig. 7 Accuracy loss and SCM loss.

tivity evaluations. Conversely, CIFAR100, particularly when evaluated with MobileNetV2, shows a low accuracy of only 31.79%, leading to an erroneous sensitivity measurement of 122.45%. The different fault sensitivity results of STL10, with around 30% accuracy, between ground truth and A-Mean are nearly complementary. Therefore, the fault sensitivity is positively correlated with accuracy.

The other reason is that fault sensitivity exhibits the accumulation and amplification of errors to some extent. The accumulation of errors is evident in the changes represented by $\Delta Acc^{'}$ and $\Delta SCM^{'}$, which reflect the variations in data before and after estimation. Since A-Mean is an evaluation method, it inherently contains some discrepancies compared to the ground truth of fault injection. However, the calculation of fault sensitivity involves using both $\Delta Acc^{'}$ and $\Delta SCM^{'}$, each carrying their own errors. The amplification of errors becomes significant because the inherent small values of $\Delta Acc^{'}$ and $\Delta SCM^{'}$ both contribute to these variations. For instance, in the ImageNet, $\Delta Acc^{'}$ ranges from 0.037% to 1.57%, while $\Delta SCM^{'}$ ranges from 0.22% to 0. 93%. These small values mean that even minor changes can cause

significant fluctuations in fault sensitivity. Despite these challenges, as depicted in Table 2, our evaluation results largely align with the ground truth, indicating that our method reliably reflects the impact of faults on avmis across different models.

4.5 Accuracy and SCM estimation improvement using max-policy for different topologies

Most DNNs are sequential, such as VGG16, VGG19, and MobileNet (shown in Fig. 4a), while the non-sequential DNNs have more complex branches, such as MobileNetV2 and ResNet50, as depicted in Fig. 4b and Fig. 4c. To characterize the fault impacts of these branch structures, the max-policy is designed to handle them and verify their effectiveness with the comparative results in Fig. 8a, using ImageNet as an example.

If the proposed A-Mean has no max-policy, all the DNN models are treated as simple sequential cases so that topology information is weakened. From Fig. 8a, it is observed that MobileNetV2 has an Acc of 70.4% and an SCM of 2.4%, while ResNet50 achieves an Acc of 74.3% and an SCM of 2% when evaluating MobileNetV2 and ResNet50 under non-topology. Our proposed max-policy approach yields similar results: MobileNetV2 has an Acc of 70.3% and an SCM of 2.4%, and ResNet50 shows an Acc of 74.2% and an SCM of 2.1%.

Comparing these results with the actual fault injection outcomes (ground truth) (Laskar et al. (2022)), where Acc values are 70% and 73.4%, and the SCM are 2.5% and 2.3% respectively, we observe that the max-policy approach provides a more accurate reflection of the true impact of faults, effectively capturing the impact on non-avmis data and accurately classified data more precisely than the original non-topology evaluations.

The reason for the effective max-policy in A-Mean is that MobileNetV2 consists of the branches in Fig. 4c, while ResNet50 includes the branches in Fig. 4b and 4c. According to the branches and the SDC with and without the max-policy, we can observe that the branches in Fig. 4c, although exhibiting a similar structure to Fig. 4a, have an additional branch without any hidden layers. As for the branch in Fig. 4b, both parts of its branches contain various hidden layers. The max-policy can effectively integrate features from both parts of Fig. 4b and 4c, resulting in favorable outcomes.

4.6 Accuracy and SCM estimation improvement using max-policy for different operations

Convolution operations dominate in DNNs, and fault occurrence in the related components certainly affects DNN Acc and SCM relatively. If our proposed method A-Mean only considers convolutions, there will be a big gap between A-Mean and ground truth (Laskar et al. (2022)), as Fig. 8b shows, using ImageNet as a case study.

Three experiments are conducted on A-Mean single convolution, with SDC_{Conv} set to 0.2, 0.25, and 0.3, respectively. The obtained ranges of Acc for the three experiments were 69.28% 74.32%, 68.99% 74.25%, and 68.71% 74.18%, and the SCM ranges from 2.01% to 2.79%, from 2.04% to 2.96%, and from 2.08% to 3.14%, respectively. Instead, if more operators, such as add, multiply, Relu, Max-pooling, etc., are considered in the enhanced A-Mean method; the estimated Acc values are from 68.99% to 74.15% , and the SCM values range from 2.10% to 2.97%. As SDC_{Conv} is set to 0.25, A-Mean single convolution performs well compared to other SDC_{Conv} value. However, multiple operations perform best among these comparisons.

An interesting phenomenon is that Acc and SCM exhibit opposite trends for DNN models; the higher Acc and the lower SCM are shown in Fig. 8. This inverse relationship results from our evaluation methodology using a unified metric SDC to assess both Acc and SCM. Acc measures the extent of correct classification, while SCM measures the extent of misclassification. As a result, the two metrics tend to move in opposite directions under the transient faults.

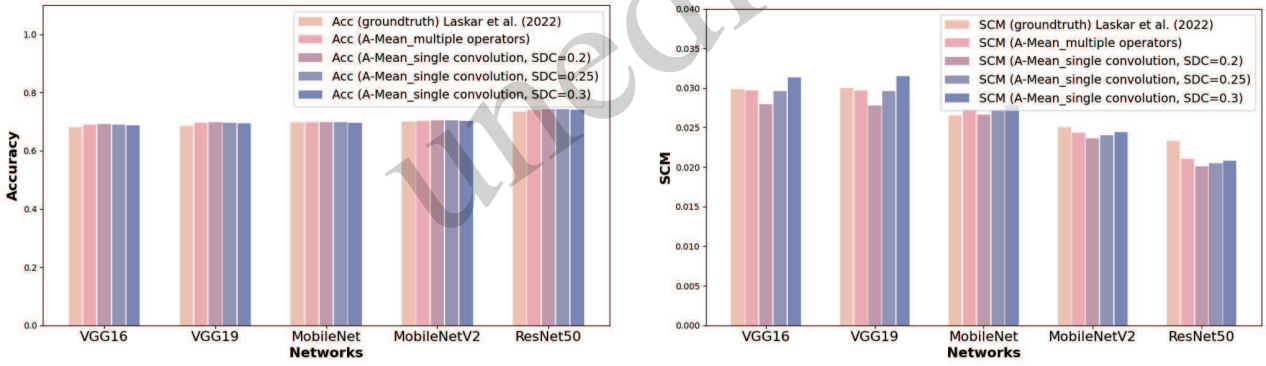
4.7 Discussion

Our proposed method, A-Mean, offers a unified, rapid, and application-level approach to fault evaluation. This method utilizes one-time dynamic execution to obtain initial values under no fault conditions. It then employs a static two-level mean value mechanism to estimate the expansion rate for each layer, followed by calculating the final expansion rate (SDC). Finally, A-Mean leverages a worst-case policy to quickly and accurately estimate the faultsâĂŹ impacts on various metrics, such as Accuracy and SCM.

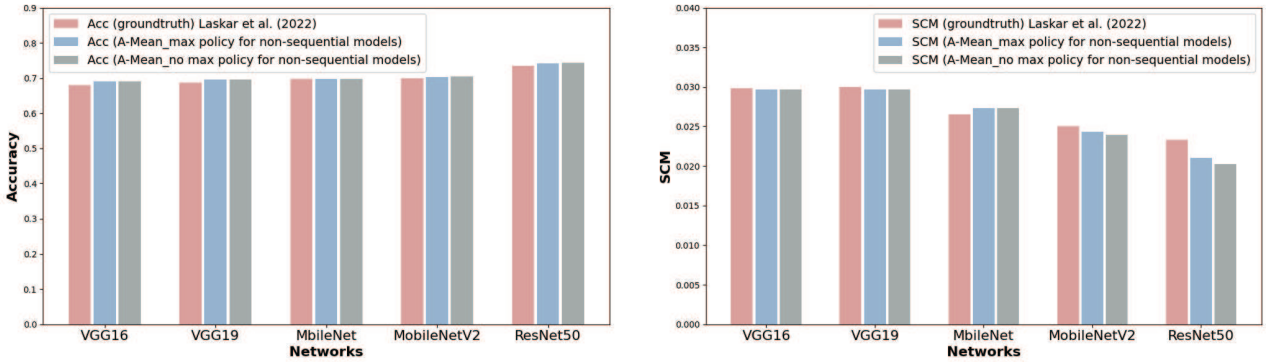
However, A-Mean also has the following limita-

Table 2 Different metrics with and without faults

Dataset	DNNs	without faults		with faults	
		Accuracy	SCM	Fault Sensitivity	
				ground truth (faults injection)	A-Mean (fault free)
ImageNet	VGG16	70.43%	2.11%	36.25%	59.49%
	VGG19	71.18%	2.04%	37.65%	58.81%
	MobileNet	70.28%	2.43%	42.31%	59.42%
	MobileNetV2	70.71%	2.21%	40.85%	59.15%
	ResNet50	74.58%	1.86%	39.50%	55.80%
CIFAR100	VGG16	64.16%	6.81%	52.70%	62.62%
	VGG19	63.82%	6.81%	38.10%	63.01%
	MobileNet	59.60%	8.28%	52.63%	66.95%
	MobileNetV2	31.79%	15.79%	52.38%	122.45%
	ResNet50	54.72%	9.84%	55.00%	72.38%
CIFAR10	VGG16	90.28%	2.07%	45.63%	44.24%
	VGG19	91.20%	2.18%	45.05%	43.63%
	MobileNet	83.45%	4.83%	30.00%	47.02%
	MobileNetV2	83.05%	4.00%	45.00%	47.80%
	ResNet50	87.65%	3.19%	37.42%	45.23%
STL10	VGG16	39.05%	27.90%	25.00%	82.32%
	VGG19	31.48%	33.35%	17.86%	95.88%
	MobileNet	32.43%	31.14%	0.00%	96.19%
	MobileNetV2	30.21%	28.33%	20.00%	78.34%
	ResNet50	30.35%	34.38%	3.03%	98.11%



(a) SCM & Accuracy estimation improvement using max-policy for different topologies



(b) SCM & Accuracy estimation improvement using max-policy for different operations

Fig. 8 SCM and Accuracy improvement on different strategies.

tions: (1) Coarse-grained insights. The method lacks detailed analysis of specific hardware vulnerabilities and may be combined with architectural estimations in future work; (2) dependence on assumptions and abstractions. This reliance might lead to an inability to capture all the nuances of real-world fault scenarios, potentially oversimplifying the fault impacts; (3) untested ultra-large-scale DNNs. A-Mean has the potential to be used for extremely large networks for its simplicity, high accuracy, and high speed.

5 Conclusion

To accelerate the assessment of the transient faultsâĂŹ impacts on DNNs, this paper presents a novel two-level analytical model, A-Mean. This proposed approach can take advantage of one-time dynamic execution and static analysis for accurate, fast, and automatic estimation using a worst-case policy. The static two-level analysis refers to inner-layer and inter-layer mean calculations. More importantly, the sequential and non-sequential DNN topologies are considered jointly, and a max-policy is used to estimate the upper bound of multiple non-sequential cases. The comprehensive results on three sequential models and two non-sequential models demonstrate that the proposed A-Mean is a fast and accurate method for characterizing the DNNâĂŹs accuracy under hardware transient faults, with 23.18 922.80x speedup, 0.13% 1.42% Accuracy loss and 0.09% 14.21% SCM loss over the advanced fault injection work. Furthermore, a novel metric fault sensitivity is defined to jointly analyze the fault impacts on Accuracy and SCM. The end-to-end automatic tool is open source for quickly and accurately evaluating the accuracy of various DNNs under transient faults.

Contributors

Jiajia Jiao and Ran Wen designed the research. Jiajia Jiao and Ran Wen processed the data. Jiajia Jiao and Ran Wen drafted the manuscript. Hong Yang helped organize the manuscript. Jiajia Jiao and Ran Wen revised and finalized the paper.

Compliance with ethics guidelines

Jia-jia JIAO, Ran WEN and Hong YANG declare that they have no conflict of interest.

References

Adam K, Mohamed II, Ibrahim Y, 2021. A selective mitigation technique of soft errors for dnn models used in healthcare applications: Densenet201 case study. *IEEE Access*, 9:65803-65823. https://doi.org/10.1109/ACCESS.2021.3076716

Ahmadilivani MH, Taheri M, Raik J, et al., 2023. Deepvigor: Vulnerability value ranges and factors for dnnsâĂŹ reliability assessment. 2023 IEEE European Test Symposium (ETS), p.1-6. https://doi.org/10.1109/ETS56758.2023.10174133

Al-haj Ahmad H, Sedaghat Y, 2022. Cafi: A configurable location-aware fault injection technique for software reliability assessment against soft errors. *Microprocessors and Microsystems*, 94:104648. https://doi.org/10.1016/j.micpro.2022.104648

BelÄĐeviÄĞ NM, StojanoviÄĞ ZN, 2022. Using voltage signals for transient fault detection on overhead lines. *International Journal of Electrical Power & Energy Systems*, 137:107824. https://doi.org/10.1016/j.ijepes.2021.107824

Camponogara Viera R, Bastos RP, Dutertre JM, et al., 2017. Method for evaluation of transient-fault detection techniques. *Microelectronics Reliability*, 76-77:68-74. https://doi.org/10.1016/j.microrel.2017.07.007

Chen Z, Narayanan N, Fang B, et al., 2020. Tensorfi: A flexible fault injection framework for tensorflow applications. 2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE), p.426-435. https://doi.org/10.1109/ISSRE5003.2020.00047

Dietrich C, Thomas TM, Mnich M, 2023. Checkpoint placement for systematic fault-injection campaigns. 2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD), p.1-9.

Du Y, 2022. The influence and application of computer technology on architectural design. 2022 8th Annual International Conference on Network and Information Systems for Computers (ICNISC), p.851-854. https://doi.org/10.1109/ICNISC57059.2022.00170

Eeckhout L, 2022. A first-order model to assess computer architecture sustainability. *IEEE Computer Architecture Letters*, 21(2):137-140. https://doi.org/10.1109/LCA.2022.3217366

Farjaminezhad R, Safari S, Eftekhari Moghadam AM, 2021a. Modeling of single/multiple-bit upset effects on logic circuits applying recurrent neural network. *Microelectronics Journal*, 117:105249. https://doi.org/10.1016/j.mejo.2021.105249

Farjaminezhad R, Safari S, Moghadam AME, 2021b. Recurrent neural networks models for analyzing single and multiple transient faults in combinational circuits. *Microelectronics Journal*, 112:104993. https://doi.org/10.1016/j.mejo.2021.104993

Gavarini G, Ruospo A, Sanchez E, 2023. Sci-fi: a smart, accurate and unintrusive fault-injector for deep neural networks. 2023 IEEE European Test Symposium (ETS), p.1-6. https://doi.org/10.1109/ETS56758.2023.10173957

Jha S, Banerjee S, Tsai T, et al., 2019. Ml-based fault injection for autonomous vehicles: A case for bayesian fault injection. 2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks

(DSN), p.112-124.
https://doi.org/10.1109/DSN.2019.00025

Jiao J, Marculescu D, Juan DC, et al., 2016. A two-level approximate model driven framework for characterizing multi-cell upsets impacts on processors. *Microelectronics Journal*, 48:7-17.
https://doi.org/10.1016/j.mejo.2015.11.011

Jooshaki M, Karimi-Arpanahi S, Millar RJ, et al., 2023. On the milp modeling of remote-controlled switch and field circuit breaker malfunctions in distribution system switch placement. *IEEE Access*, 11:40905-40915.
https://doi.org/10.1109/ACCESS.2023.3268993

Jung J, Ko Y, So H, et al., 2022. Root cause analysis of soft-error-induced failures from hardware and software perspectives. *Journal of Systems Architecture*, 130:102652.
https://doi.org/10.1016/j.sysarc.2022.102652

Laskar S, Rahman MH, Zhang B, et al., 2022. Characterizing deep learning neural network failures between algorithmic inaccuracy and transient hardware faults. 2022 IEEE 27th Pacific Rim International Symposium on Dependable Computing (PRDC), p.54-67.
https://doi.org/10.1109/PRDC55274.2022.00020

Li G, Hari SKS, Sullivan M, et al., 2017. Understanding error propagation in deep learning neural network (dnn) accelerators and applications. SC17: International Conference for High Performance Computing, Networking, Storage and Analysis, p.1-12.

Li P, Zhen L, Li X, et al., 2021. Radiation hardness assurance of single event effects on components for space application. 2021 4th International Conference on Radiation Effects of Electronic Devices (ICREED), p.1-6.
https://doi.org/10.1109/ICREED52909.2021.9588712

Liang J, Li Y, Yin G, et al., 2023. A mas-based hierarchical architecture for the cooperation control of connected and automated vehicles. *IEEE Transactions on Vehicular Technology*, 72(2):1559-1573.
https://doi.org/10.1109/TVT.2022.3211733

Mukherjee S, 2008. Architecture design for soft errors. Morgan Kaufmann.
https://doi.org/10.1016/B978-0-12-369529-1.X5001-0

Papadimitriou G, Gizopoulos D, 2021. Demystifying the system vulnerability stack: Transient fault effects across the layers. 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA), p.902-915.
https://doi.org/10.1109/ISCA52012.2021.00075

Papadimitriou G, Gizopoulos D, Dixit HD, et al., 2023. Silent data corruptions: The stealthy saboteurs of digital integrity. 2023 IEEE 29th International Symposium on On-Line Testing and Robust System Design (IOLTS), p.1-7.
https://doi.org/10.1109/IOLTS59296.2023.10224870

Ping L, Tan J, Yan K, 2020. Sern: Modeling and analyzing the soft error reliability of convolutional neural networks. Proceedings of the 2020 on Great Lakes Symposium on VLSI, New York, NY, USA, p.445âĂŞ450.
https://doi.org/10.1145/3386263.3406938

Raj S, Singh V, Rajalwal NK, et al., 2020. Reliability prediction of a distribution protection scheme using markov model. 2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), p.868-872.
https://doi.org/10.1109/ICRITO48877.2020.9197804

Ramzanpour M, Ludwig SA, 2020. Association rule mining based algorithm for recovery of silent data corruption in convolutional neural network data storage. 2020 IEEE Symposium Series on Computational Intelligence (SSCI), p.3057-3064.
https://doi.org/10.1109/SSCI47803.2020.9308545

Ruospo A, Gavarini G, Bragaglia I, et al., 2022. Selective hardening of critical neurons in deep neural networks. 2022 25th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS), p.136-141.
https://doi.org/10.1109/DDECS54261.2022.9770168

Ruospo A, Sanchez E, Luza LM, et al., 2023. A survey on deep learning resilience assessment methodologies. *Computer*, 56(2):57-66.
https://doi.org/10.1109/MC.2022.3217841

Sangchoolie B, Pattabiraman K, Karlsson J, 2017. One bit is (not) enough: An empirical study of the impact of single and multiple bit-flip errors. 2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), p.97-108.
https://doi.org/10.1109/DSN.2017.30

Shi Y, Tong Z, Zhang Z, et al., 2023. Research on jamming decision technology based on hierarchical architecture. 2023 4th International Conference on Electronic Communication and Artificial Intelligence (ICECAI), p.52-56.
https://doi.org/10.1109/ICECAI58670.2023.10176793

Sun R, Qiu P, Lyu Y, et al., 2021. Lightning: Striking the secure isolation on gpu clouds with transient hardware faults. *ArXiv*, abs/2112.03662.
https://api.semanticscholar.org/CorpusID:244920967

Taheri M, Ahmadilivani MH, Jenihhin M, et al., 2023. Appraiser: Dnn fault resilience analysis employing approximation errors. 2023 26th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS), p.124-127.
https://doi.org/10.1109/DDECS57882.2023.10139468

Tan J, Ping L, Wang Q, et al., 2023a. Saca-avf: A quantitative approach to analyze the architectural vulnerability factors of cnn accelerators. *IEEE Transactions on Computers*, 72(11):3042-3056.
https://doi.org/10.1109/TC.2023.3283685

Tan J, Wang Q, Yan K, et al., 2023b. Saca-fi: A microarchitecture-level fault injection framework for reliability analysis of systolic array based cnn accelerator. *Future Gener Comput Syst*, 147(C):251âĂŞ264.
https://doi.org/10.1016/j.future.2023.05.009

Venkatesha S, Parthasarathi R, 2022. One shot system based reliability modelling and analysis for low-cost fault-tolerant computing system comprising of one instruction cores. 2022 International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON), p.1-9.
https://doi.org/10.1109/SMARTGENCON56628.2022.10084187

Yao G, Yang W, Liu H, 2022. The design of the operational monitoring system of the state grid on the internet based on the computer architecture. 2022 World Automation Congress (WAC), p.608-613.
https://doi.org/10.23919/WAC55640.2022.9934282

Zheng Y, Feng Z, Hu Z, et al., 2021. Mindfi: A fault injection tool for reliability assessment of mindspore applicacions. 2021 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), p.235-238.
https://doi.org/10.1109/ISSREW53611.2021.00068

Zhou Q, Luo Z, Ouyang X, et al., 2020. Analysis of the influence of optical fiber layout on the internal electric field of power transformer. 2020 IEEE International Conference on High Voltage Engineering and Application (ICHVE), p.1-4.
https://doi.org/10.1109/ICHVE49031.2020.9279811