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 deep neural networks 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 May 12, 2025

Abstract: Hardware transient faults are proven to have a significant impact on deep neural networks (DNNs), whose safety-critical misclassification (SCM) in autonomous vehicles, healthcare, and space applications is increased up to four times. 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, using the silent data corruption (SDC) rate of basic operations (such as convolution, addition, multiply, ReLU, and max-pooling) and a static two-level mean calculation mechanism to rapidly compute the overall SDC rate, for estimating the general classification metric accuracy and application-specific metric 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 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 prompt evaluation of transient faults on diverse DNNs. Meanwhile, a novel metric "fault sensitivity" is defined to characterize the variation of transient fault-induced higher SCM and lower accuracy. The comparative results with a state-of-the-art fault injection method TensorFI+ on five DNN models and four datasets show that our proposed estimation method A-Mean achieves up to 922.80 times speedup, with just 4.20% SCM loss and 0.77% accuracy loss on average. The artifact of A-Mean is publicly available at https://github.com/breatrice321/A-Mean.

Key words: Analytical model; Deep neural networks; Hardware transient faults; Fast evaluation; Automatic evaluation tool

https://doi.org/10.1631/FITEE.2400547 **CLC number:** TP391

1 Introduction

Transient faults (Farjaminezhad et al., 2021b; Papadimitriou and Gizopoulos, 2021; Belčević and Stojanović, 2022) primarily result from inherent circuit malfunctions (Jooshaki et al., 2023), such as external radiation (Jiao et al., 2016; Al-haj Ahmad and Sedaghat, 2022) and internal electrical interference (Zhou et al., 2020; Jung et al., 2022). These transient faults often occur randomly and cause 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 PW

‡ Corresponding author

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

ORCID: Jiajia JIAO, https://orcid.org/0000-0003-3680-787X

© Zhejiang University Press 2025

et al., 2021), transient faults could lead to immeasurable loss of life and property. In recent years, deep neural networks (DNNs) (Chen et al., 2020; Laskar et al., 2022; Tan et al., 2023a) have been adopted in 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 (Farjaminezhad et al., 2021a; Ahmadilivani et al., 2023; Dietrich et al., 2023).

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 fault impacts on DNNs, such as SERN (Ping et al., 2020), APPRAISER (Taheri et al., 2023), DeepVigor (Ahmadilivani et al., 2023), and 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 advantage of high 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, there are still two problems to be solved. One is that some dynamic information of the images inputted 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 automatic methodology A-Mean for fast evaluation of the accuracy of DNNs. This approach combines dynamic information from one-time fault-free execution with static information from a fast analytical model for 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 advantages of one-time fault-free dynamic information, a static two-level mean calculation model, and a worst-case policy to effectively capture the inner-layer (data type and operator) and inter-layer (input feature map, depth, and topology) fault impacts. It can achieve an accuracy up to 99.23% on average and up to 922.80 times speedup over the state-of-the-art fault injection method.

2. We conduct some experiments and compare our A-Mean with single-convolution silent data corruption (SDC) and no max-policy. As we implement three groups of $\mathrm{SDC}_{\mathrm{Conv}}$ to replace other $\mathrm{SDC}_{\mathrm{op}}$, 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 uses the model's 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 s. Meanwhile, a new metric, fault sensitivity, is proposed for estimating transient

fault impacts on the variation of accuracy and SCM. We have made the artifact of A-Mean publicly available at https://github.com/breatrice321/A-Mean.

2 Background and related works

2.1 Transient faults impact on DNNs

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

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, as 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 (Liang et al., 2023; Shi 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, gives DNNs some fault tolerance (Camponogara Viera et al., 2017; Sun et al., 2021; Jung et al., 2022). 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 under TensorFI+ (Laskar et al., 2022). Therefore, we can observe that DNNs exhibit

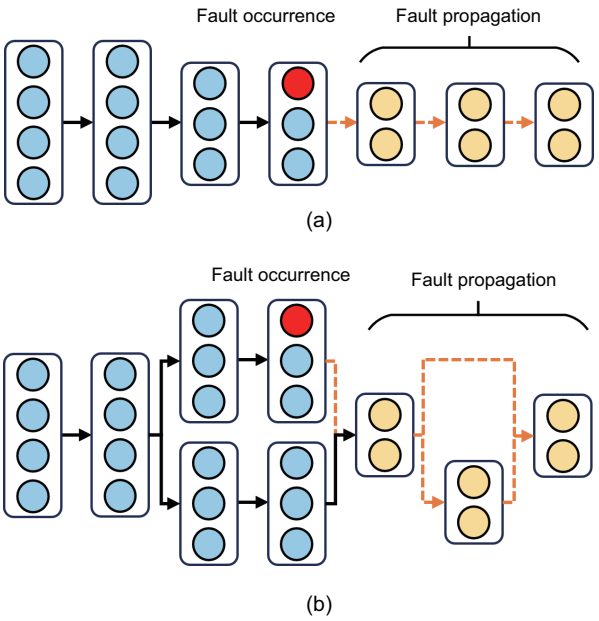

Fig. 1 Fault occurrence and propagation in DNNs: (a) sequential models; (b) non-sequential models (References to color refer to the online version of this figure)

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 SCM, as these faults can result in catastrophic consequences. Hence, it is essential to accurately evaluate the impact of faults on these applications.

2.2 Fault models

To better understand the impact of faults, this paper provides a detailed description of the fault injection process, including fault time, fault location, and fault type.

1. 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 works (Chen et al., 2020; Laskar et al., 2022).

2. Fault location. Transient faults probably occur in the inputs, 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 (Chen et al., 2020; Laskar et al., 2022).

3. Fault type. The increasing multi-bit upset has only an 8% probability of causing SDC (Sangchoolie et al., 2017); Sangchoolie et al. (2017) showed that multi-bit upset-induced errors exhibit similar error propagation patterns and SDC rates 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 for hardware transient faults on DNNs

Hardware transient faults make the reliable design of computer architecture increasingly significant (Raj et al., 2020; Du, 2022; Eeckhout, 2022; Venkatesha and Parthasarathi, 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) proposed 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 are time-consuming, require performing many experiments, and are limited to specific hardware architectures. Thus, it is difficult to satisfy the real-time requirements and achieve good generalization to diverse safety-critical applications.

A fast fault-free analysis method 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 convolutional neural networks (CNNs) using only a small number of CNN parameters. Ahmadilivani et al. (2023) presented a fine-grain, metric-oriented evaluation method, DeepVigor, which 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 accelerator's 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 fault impacts on DNN inaccuracy. The proposed approach uses one-time fault-free execution to capture the basic no-fault SCM information and then the information is combined with a two-level analytical model for SCM evaluation under transient faults. More importantly, an end-to-end automatic tool is implemented in this well-generalized work for fast and accurate evaluation.

2.4 Reliability metrics: SDC rate and SCM

SDC is an intricate challenge in computer systems and appears 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 unveiled only 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 can be calculated as the ratio of the number of SDC results to the total number of executions using fault injection.

SCM is a specific metric for measuring the misclassifications of safety-critical data in DNNs, and we redefine 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 is supposed to be applied will be cancelled, which will cause significant safety hazards (avmis). Therefore, A-Mean is proposed to evaluate this metric on diverse DNNs. 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.

$$\mathrm{Acc}=\frac{n_{\mathrm{correct}}}{n_{\mathrm{avmis}}+n_{\mathrm{non\text{-}avmis}}+n_{\mathrm{correct}}}, \tag{1}$$

$$\mathrm{SCM}=\frac{n_{\mathrm{avmis}}}{n_{\mathrm{avmis}}+n_{\mathrm{non\text{-}avmis}}+n_{\mathrm{correct}}}. \tag{2}$$

3 Proposed analytical model A-Mean

3.1 Overall framework

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

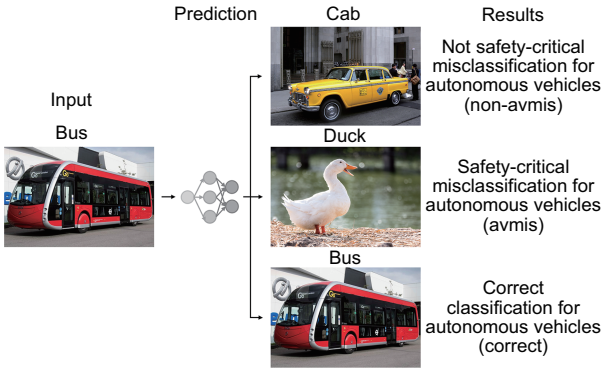

Fig. 2 Definition of non-avmis, avmis, and correct classifications

1. One-time 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 worsen SCM of the classification. Therefore, it is necessary to obtain the basic Acc and SCM values. This procedure requires only one-time dynamic execution to obtain the classification results.

2. Static two-level mean calculation. After the one-time 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 as 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 operation types, input feature maps, the depth of networks, and topologies. Therefore, this paper considers these factors in a joint way. As for 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 the worst-case policy. Based on the Acc and SCM from the one-time 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 Eqs. (3) and (4). $\mathrm{Acc}_{\mathrm{no\_fault}}$ represents the accuracy without faults, and $\mathrm{InAcc}_{\mathrm{no\_fault}}$ represents the inaccuracy without faults. The sum of these two values is equal to 1. $\mathrm{nonSCM}_{\mathrm{no\_fault}}$ represents the nonSCM value without faults, while $\mathrm{SCM}_{\mathrm{no\_fault}}$ is the SCM value without faults. The sum of $\mathrm{nonSCM}_{\mathrm{no\_fault}}$ and $\mathrm{SCM}_{\mathrm{no\_fault}}$ is 1.

In the worst case, transient faults do not alter the misclassification results of $\mathrm{InAcc}_{\mathrm{no\_fault}}$, nor do they affect $\mathrm{SCM}_{\mathrm{no\_fault}}$. Meanwhile, some of the

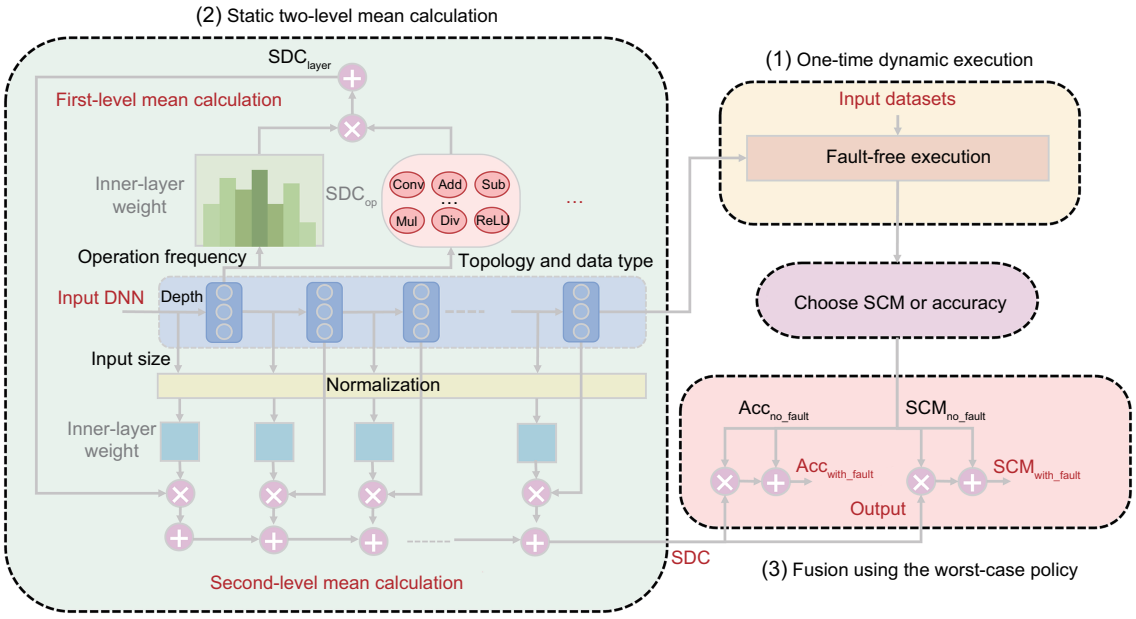

Fig. 3 Comparison of different methods in terms of outlier detection accuracy

$\mathrm{Acc}_{\mathrm{no\_fault}}$ values become $\mathrm{InAcc}_{\mathrm{no\_fault}}$ using the overall SDC as the expansion rate in Eq. (3).

However, compared with the influence of SDC on $\mathrm{Acc}_{\mathrm{no\_fault}}$, $\mathrm{nonSCM}_{\mathrm{no\_fault}}$ needs to be considered in two respects. As illustrated in Eqs. (1) and (2), SCM focuses exclusively on avmis, while Acc considers only 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 $1/(2+\varepsilon)$ $(\varepsilon>0)$ correct cases will turn into avmis, with the remaining proportion turning into the non-avmis category.

$$\begin{aligned}&\mathrm{Acc}_{\mathrm{with\_fault}}\\&=1-\left(\mathrm{Acc}_{\mathrm{no\_fault}}\cdot\mathrm{SDC}+\mathrm{InAcc}_{\mathrm{no\_fault}}\right),\end{aligned}\tag{3}$$

$$\begin{aligned}&\mathrm{SCM}_{\mathrm{with\_fault}}\\&=\left[\frac{1}{2}(\mathrm{nonSCM}_{\mathrm{no\_fault}}-\mathrm{Acc}_{\mathrm{no\_fault}})\right.\\&\left.+\frac{1}{2+\varepsilon}\mathrm{Acc}_{\mathrm{no\_fault}}\right]\cdot\mathrm{SDC}+\mathrm{SCM}_{\mathrm{no\_fault}}.\end{aligned}\tag{4}$$

3.2 Details in the static two-level mean calculation 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 $\mathrm{SDC}_{\mathrm{layer}}$ 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 $\mathrm{SDC}_{\mathrm{op}}$ as the fundamental infrastructure and the operation frequency as their corresponding inner-layer weight $\mathrm{Weight}_{\mathrm{op}}$ to compute $\mathrm{SDC}_{\mathrm{layer}}$ using Eq. (5):

$$\mathrm{SDC}_{\mathrm{layer}_i}=\sum_{j=0}^{k-1}\mathrm{Weight}_{\mathrm{op}_j}\cdot\mathrm{SDC}_{\mathrm{op}_j}.\tag{5}$$

The weight of the j^{th} operation $\mathrm{Weight}_{\mathrm{op}_j}$ is the ratio of the operation frequency of the j^{th} operation to the operation frequency of the total 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 and

rectified linear unit (ReLU). The physical implementation determines operations' area consumption and fault occurrence probability. It can be configured to be suitable for different implementations by changing the weight easily.

$$\begin{cases}\text{Weight}_{\text{op}_j}=\frac{N_{\text{op}_j}}{\sum_{j=0}^{k-1}N_{\text{op}_j}},\\ \sum_{j=0}^{k-1}\text{Weight}_{\text{op}_j}=1.\end{cases} \tag{6}$$

In this paper, 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 can also be estimated using other available evaluation methods.

Table 1 SDC_{op} of different operators

Operation	SDC_{op}	
	A-Mean	Chen et al. (2020)
Convention	0.250	0.22
Addition	0.125	0.10
Subtraction	0.125	0.02
Multiplication	0.125	0.15
Division	0.125	0.15
ReLU	0.375	0.30
Max-pooling	0.250	0.35
Average pooling	1.000	1.00

2. Second-level mean calculation for joint layers. The output of the first-level mean value for each layer $\text{SDC}_{\text{layer}}$ is the input to the second level to characterize the correlation between different layers in Eq. (7):

$$\text{SDC}=\sum_{i=0}^{n-1}\text{Weight}_{\text{layer}_i}\cdot\text{SDC}_{\text{layer}_i}. \tag{7}$$

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):

$$\begin{cases}\text{Weight}_{\text{layer}_i}=\frac{W_iH_iC_i}{\sum_{i=0}^{n-1}W_iH_iC_i},\\ \sum_{i=0}^{n-1}\text{Weight}_{\text{layer}_i}=1.\end{cases} \tag{8}$$

The depth of the network is taken into account in A-Mean. From Li GP et al. (2017), we find 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 are more likely to propagate to the subsequent layers. In our work, we select inter-layer weights and use 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 the 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. (9). Additionally, as indicated in Eq. (10), 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 $\log(\text{depth}_{\text{layer}_i})+\varepsilon'$. This approach helps moderate the effect of depth while accounting for differences across layers.

$$\text{Weight}_{\text{layer}_i}\longleftarrow\frac{\text{Weight}_{\text{layer}_i}}{\text{influence}_{\text{layer}_i}}, \tag{9}$$

$$\text{influence}_{\text{layer}_i}=\text{depth}_{\text{layer}_i}-\left(\log(\text{depth}_{\text{layer}_i})+\varepsilon'\right). \tag{10}$$

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} values. For example, as shown in Table 1, the second column represents the SDC calculated using our method. The third column shows the data from Chen et al. (2020), derived from fault injection experiments measuring the SDC for each operation. Based on two different SDC_{op} values, we calculate SDC rates for DNNs, such as VGG16, to be 0.0205 (A-Mean) and 0.0182 (Chen et al., 2020). Therefore, suitable SDC_{op} values are supposed to be provided according to model operations and other influence factors.

Previous works (Li GP 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} values

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 GP et al., 2017; Zheng et al., 2021). Therefore, $\mathrm{SDC}_{\mathrm{basic}}$ is defined as the ratio of the number of exponent bits of the higher-order part to the total number of bits, as shown in Eq. (11):

$$\mathrm{SDC}_{\mathrm{basic}}=\frac{n_{\mathrm{high\text{-}order\_exponent}}}{n_{\mathrm{sign}}+n_{\mathrm{exponent}}+n_{\mathrm{tail}}}. \tag{11}$$

As for float32, the exponential part is 8-bit. The highest 4 bits of the exponential part are assumed to influence SDC significantly. Therefore, the basic value of $\mathrm{SDC}_{\mathrm{basic}}$ is set as $4/32=1/8=0.125$. 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. Add is often used to connect data, such as residual connections, Mul is often used in weight updates, activation functions, and other aspects, and Sub and Div usually appear in data normalization. These simple operations are fundamental for more complex operations. In this paper, $\mathrm{SDC}_{\mathrm{Add}}$, $\mathrm{SDC}_{\mathrm{Sub}}$, $\mathrm{SDC}_{\mathrm{Mul}}$, and $\mathrm{SDC}_{\mathrm{Div}}$ are set to $\mathrm{SDC}_{\mathrm{basic}}$ as mentioned above.

2. Convolution (Conv) operation. The essence of Conv calculation is Mul and Add. Therefore, we take the sum of $\mathrm{SDC}_{\mathrm{Add}}$ and $\mathrm{SDC}_{\mathrm{Mul}}$ as $\mathrm{SDC}_{\mathrm{Conv}}$, so $\mathrm{SDC}_{\mathrm{Conv}}=\mathrm{SDC}_{\mathrm{Mul}}+\mathrm{SDC}_{\mathrm{Add}}$.

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

$$\mathrm{ReLU}=\max(0,wx+b). \tag{12}$$

ReLU consists mainly of two parts: one is 0 ($x<0$), and the other is $wx+b$ ($x>0$). 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 $\mathrm{SDC}_{\mathrm{ReLU\_part}}=0.5$ as $x<0$. When $x>0$, this part of operation consists of two parts (Mul and Add), and this part $\mathrm{SDC}_{\mathrm{ReLU\_part}}=\mathrm{SDC}_{\mathrm{basic}}\times 2$. Therefore, $\mathrm{SDC}_{\mathrm{ReLU}}$ is set according to Eq. (14), which uses the mean value of two parts to fully consider the impact of the bit upset.

$$\mathrm{SDC}_{\mathrm{ReLU\_part}}=\begin{cases}\mathrm{SDC}_{\mathrm{basic}}\times 2, & x>0,\\ 0.5, & x<0,\end{cases} \tag{13}$$

$$\mathrm{SDC}_{\mathrm{ReLU}}=\frac{\mathrm{SDC}_{\mathrm{basic}}\times 2+0.5}{2}. \tag{14}$$

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, $\mathrm{SDC}_{\mathrm{avg\_pool}}=1$. 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, $\mathrm{SDC}_{\mathrm{max\_pool}}=\mathrm{SDC}_{\mathrm{Conv}}$.

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

A max-policy is designed to handle these branches in Figs. 4b and 4c for non-sequential DNN models. To guarantee the reliability, we compute the maximum $\mathrm{SDC}_{\mathrm{layer}}$ of multiple layers 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 max-policy. $\mathrm{SDC}_{\mathrm{max}}$ in Fig. 4b is the maximum $\mathrm{SDC}_{\mathrm{layer}}$ value of $p+q$ layers in two branches. As Fig. 4c shows, the maximum value of $\{\mathrm{SDC}_{\mathrm{br0\_0}}, \mathrm{SDC}_{\mathrm{br0\_1}}, \ldots, \mathrm{SDC}_{\mathrm{br0\_}m-1}\}$ is set to $\mathrm{SDC}_{\mathrm{max}}$. After the parallel computation of different branches, $\mathrm{SDC}_{\mathrm{max}}$ is used to replace all the $\mathrm{SDC}_{\mathrm{layer}}$ values 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 static two-level mean calculation mechanism for sequential structures and max-policy for branches in non-sequential structures. Therefore, the proposed A-Mean considers different kinds of topologies' impacts on the fault propagation in DNNs for accurate $\mathrm{Acc}_{\mathrm{with\_fault}}$ and $\mathrm{SCM}_{\mathrm{with\_fault}}$ estimation.

3.5 End-to-end automatic tool

To use our A-Mean for fast and accurate evaluation conveniently, we develop 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, operation type, 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 $\mathrm{Acc}_{\mathrm{with\_fault}}$ and $\mathrm{SCM}_{\mathrm{with\_fault}}$ are 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 s. It is highly efficient and suitable for real-time requirements of diverse safety-critical applications.

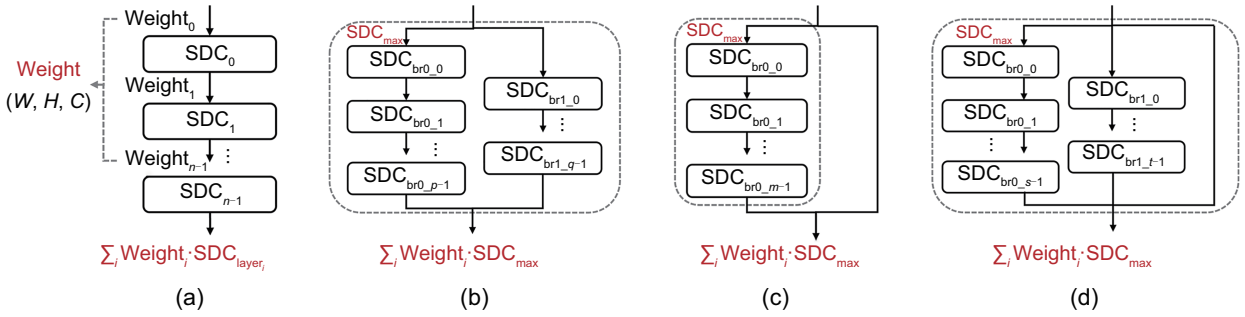

Fig. 4 Various DNN topologies: (a) sequential structure; (b) branch topology 1 (both branches have hidden layers); (c) branch topology 2 (one branch contains hidden layers); (d) branch topology 3 (some branches have hidden layers and some do not)

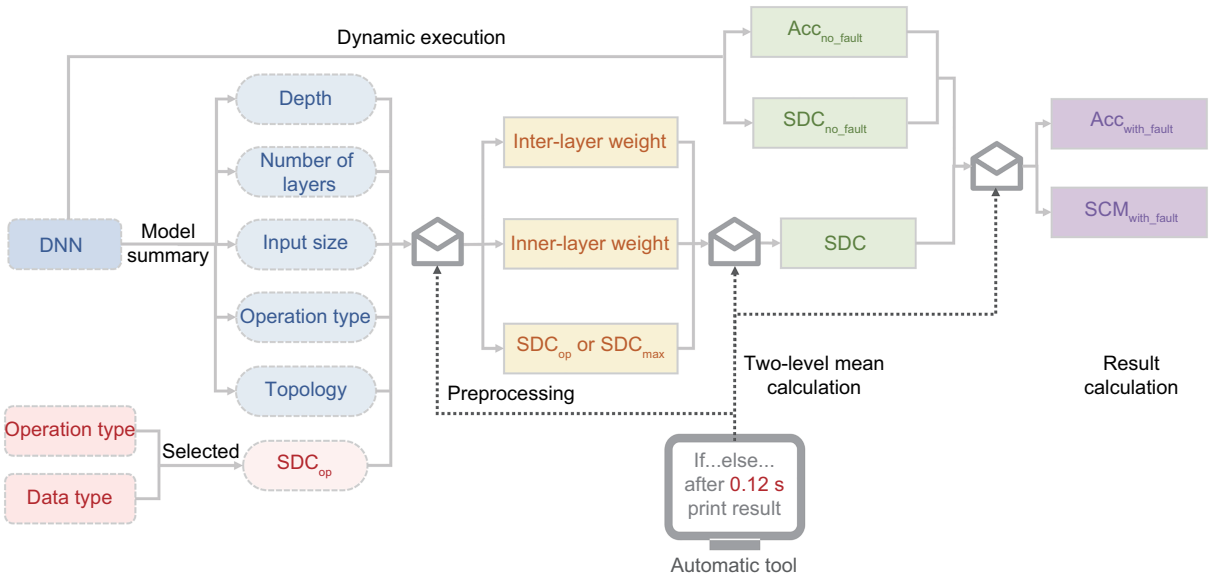

Fig. 5 End-to-end automatic tool

3.6 Validating A-Mean by fault injection

To verify the effectiveness of the proposed A-Mean, we conduct 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 types of metrics from three perspectives (speed, accuracy, and SCM) to validate A-Mean, which include the speedup of estimation, SCM loss, accuracy loss, and fault sensitivity.

Speedup of estimation reflects the acceleration of our automatic evaluation method over the SOTA fault injection method, as shown in Eq. (15). $\text{Runtime}_{\text{A-Mean}}$ is the total runtime of A-Mean, including the time of one-time dynamic execution and the estimation time, while $\text{Runtime}_{\text{TensorFI+}}$ represents the total time of multiple fault injections using TensorFI+.

$$\text{Speedup} = \frac{\text{Runtime}_{\text{TensorFI+}}}{\text{Runtime}_{\text{A-Mean}}}. \tag{15}$$

$\text{Acc}_{\text{A-Mean}}$ and $\text{SCM}_{\text{A-Mean}}$ are estimated by the proposed A-Mean method using one-time dynamic execution, while $\text{Acc}_{\text{ground-truth}}$ and $\text{SCM}_{\text{ground-truth}}$ represent the results of N random fault injections. ΔAcc and ΔSCM are calculated to give the absolute deviation between the 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 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 and with faults, as described in Eq. (18). This approach helps us observe how faults affect safety-critical aspects of the model.

$$\begin{cases} \Delta\text{Acc} = \left|\text{Acc}_{\text{A-Mean}} - \text{Acc}_{\text{ground-truth}}\right|, \\ \text{Accuracy loss} = \frac{\Delta\text{Acc}}{\text{Acc}_{\text{ground-truth}}}, \end{cases} \tag{16}$$

$$\begin{cases} \Delta\text{SCM} = \left|\text{SCM}_{\text{A-Mean}} - \text{SCM}_{\text{ground-truth}}\right|, \\ \text{SCM loss} = \frac{\Delta\text{SCM}}{\text{SCM}_{\text{ground-truth}}}, \end{cases} \tag{17}$$

$$\begin{cases} \Delta\text{SCM}' = \left|\text{SCM}_{\text{with\_fault}} - \text{SCM}_{\text{no\_fault}}\right|, \\ \Delta\text{Acc}' = \left|\text{Acc}_{\text{with\_fault}} - \text{Acc}_{\text{no\_fault}}\right|, \\ \text{Fault sensitivity} = \frac{\Delta\text{SCM}'}{\Delta\text{Acc}'}. \end{cases} \tag{18}$$

4 Results and analysis

4.1 Experimental setup and configuration

The experimental configuration is shown as follows: all the experiments in this paper are run on our private server with Ubuntu Linux 18.04.6 LTS on an Intel Xeon® Silver 4210 2.2 GHz processor with 125.5 GB of DDR4 memory. The datasets include ImageNet, CIFAR100 and CIFAR10 with 10 000 images, and STL10 with 8000 images.

To validate the effectiveness of A-Mean through fault injection, all the data from four datasets are used 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

Runtime of the proposed A-Mean is shorter than that of TensorFI+ (Laskar et al., 2022), as shown in Fig. 6. TensorFI+ based on fault injection consumes up to 98 h, while A-Mean achieves a one to two orders of magnitude improvement over TensorFI+, 21.85–922.80 times speedup in four different datasets. Due to one-time dynamic execution in A-Mean instead of multiple executions in FI, the runtime is reduced dramatically to around 0.11–0.50 h. Additionally, the static estimation of the two-level mean calculation and the worst-case policy, along with the network preprocessing required for static estimation, requires less than 0.12 s and has negligible impact. Therefore, A-Mean is more suitable for fast estimation of transient fault impact on DNN inaccuracy.

Furthermore, the speedup of A-Mean under different networks varies significantly as shown in Fig. 6. This is mainly due to the structural differences among networks, particularly in non-sequential

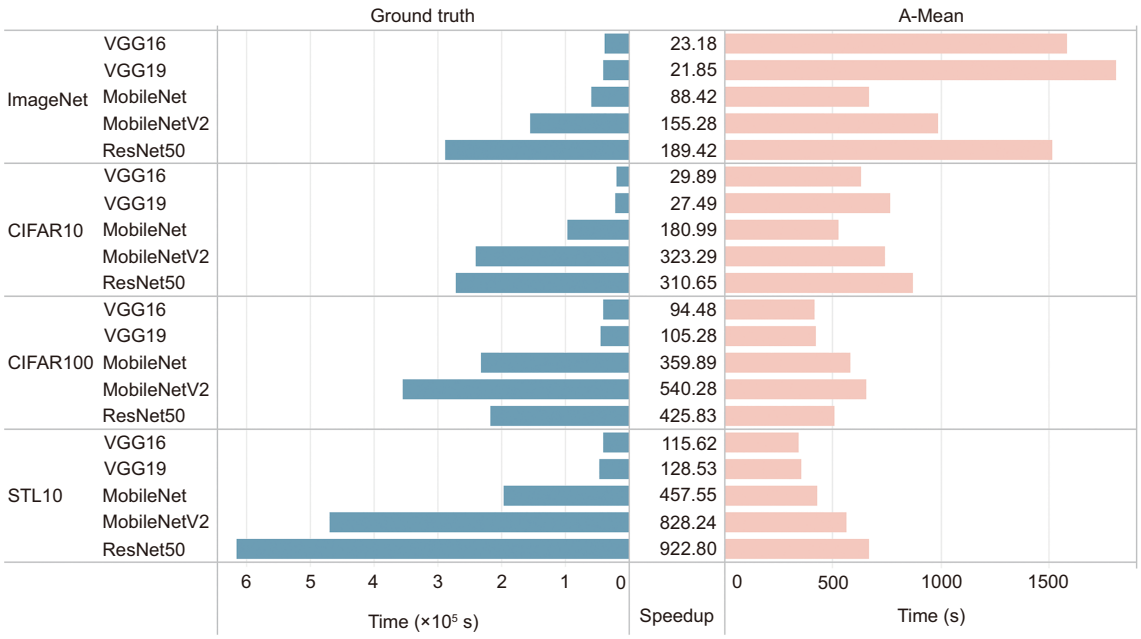

Fig. 6 Speedup between A-Mean and TensorFI+ (ground truth)

networks such as MobileNetV2 and ResNet50. The speedups achieved by these two models range from 155.28 to 922.80 times across four datasets and rank the highest across different networks on the same dataset. In contrast, sequential networks such as VGG16, VGG19, and MobileNet exhibit relatively low speedups, ranging from 21.85 to 457.55 times. This phenomenon can be attributed to the fact that non-sequential DNNs, due to their deeper layers and more complex structures, necessitate longer and more time-consuming fault injection procedures compared to simple sequential networks. Therefore, the proposed A-Mean performs better in terms of speed advantage in complex DNN models.

4.3 Estimation accuracy using SCM loss and accuracy loss

The estimation results of $\mathrm{Acc}_{\mathrm{with\_fault}}$ and $\mathrm{SCM}_{\mathrm{with\_fault}}$ by our proposed A-Mean and SOTA fault injection method TensorFI+ (Laskar et al., 2022) are shown in Fig. 7. TensorFI+ provides $\mathrm{Acc}_{\mathrm{ground\text{-}truth}}$ and $\mathrm{SCM}_{\mathrm{ground\text{-}truth}}$ (this paper reproduces the fault injection work to align with the experiments), while our A-Mean computes the estimated $\mathrm{Acc}_{\mathrm{A\text{-}Mean}}$ and $\mathrm{SCM}_{\mathrm{A\text{-}Mean}}$.

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 consistently low, demonstrating that A-Mean can achieve estimation accuracy comparable to fault injection. As shown in Fig. 7b, the SCM loss values range from 0.42% to 9.85% on ImageNet, 0.09% to 8.00% on CIFAR100, 2.96% to 14.21% on CIFAR10, and 0.46% to 2.13% on STL10, across the five DNN models.

Consequently, our A-Mean method can effectively accelerate the transient fault 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 calculation mechanism considers the input feature map, depth, operator type, and topology 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 $\Delta\mathrm{SCM}'$ to $\Delta\mathrm{Acc}'$ without and with faults. It can directly reflect the fault impacts on increasing SCM over decreasing accuracy.

As illustrated in Table 2, the fault sensitivity of

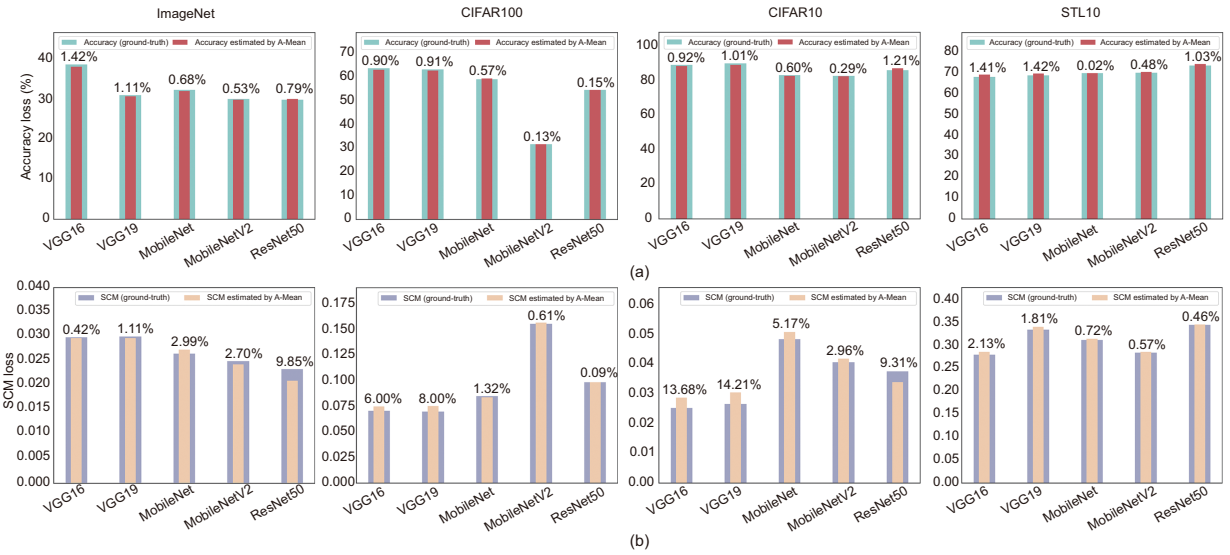

Fig. 7 Accuracy loss (a) and SCM loss (b) on four datasets

Table 2 Different metrics with and without faults

Dataset	DNN	Acc_{no_fault} (%)	SCM_{no_fault} (%)	Fault sensitivity with faults	
				Ground truth	A-Mean
ImageNet	VGG16	70.43	2.11	0.3625	0.5949
	VGG19	71.18	2.04	0.3765	0.5881
	MobileNet	70.28	2.43	0.4231	0.5942
	MobileNetV2	70.71	2.21	0.4085	0.5915
	ResNet50	74.58	1.86	0.3950	0.5580
CIFAR100	VGG16	64.16	6.81	0.5270	0.6262
	VGG19	63.82	6.81	0.3810	0.6301
	MobileNet	59.60	8.28	0.5263	0.6695
	MobileNetV2	31.79	15.79	0.5238	1.2245
	ResNet50	54.72	9.84	0.5500	0.7238
CIFAR10	VGG16	90.28	2.07	0.4563	0.4424
	VGG19	91.20	2.18	0.4505	0.4363
	MobileNet	83.45	4.83	0.3000	0.4702
	MobileNetV2	83.05	4.00	0.4500	0.4780
	ResNet50	87.65	3.19	0.3742	0.4523
STL10	VGG16	39.05	27.90	0.2500	0.8232
	VGG19	31.48	33.35	0.1786	0.9588
	MobileNet	32.43	31.14	0	0.9619
	MobileNetV2	30.21	28.33	0.2000	0.7834
	ResNet50	30.35	34.38	0.3030	0.9811

fault injection (ground truth) is compared with that of A-Mean. The fault sensitivity estimated by A-Mean ranges from 0.5580 to 0.5949 on ImageNet, 0.6262 to 1.2245 on CIFAR100, 0.4363 to 0.4780 on CIFAR10, and 0.7834 to 0.9811 on STL10. In comparison, the ground-truth fault sensitivity value ranges from 0.3625 to 0.4231 on ImageNet, 0.3810 to 0.5500 on CIFAR100, 0.3000 to 0.4563 on CIFAR10, and 0 to 0.2500 on STL10. These comparisons reveal variations in assessment outcomes to some extent. For example, ImageNet (first group) demonstrates relatively high evaluation results, while CIFAR100 (second group) shows anomalous data with a value of 1.22. Meanwhile, CIFAR10 (third group) yields more accurate fault sensitivity results. However, STL10 (fourth group) presents the complementary fault sensitivity between ground-truth and A-Mean.

One primary reason for these discrepancies

lies in different classification accuracy levels of the datasets, as highlighted in Table 2. CIFAR10 exhibits substantially the highest accuracy compared to the other three datasets, resulting in more precise sensitivity evaluations. Conversely, CIFAR100, particularly when evaluated with MobileNetV2, shows a low accuracy of only 31.79%, leading to an erroneous sensitivity measurement of 1.2245. 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 accumulation and amplification of errors to some extent. The accumulation of errors is evident in the changes represented by $\Delta\mathrm{Acc}'$ and $\Delta\mathrm{SCM}'$, 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 $\Delta\mathrm{Acc}'$ and $\Delta\mathrm{SCM}'$, each carrying its own errors. The amplification of errors becomes significant because the inherent small values of $\Delta\mathrm{Acc}'$ and $\Delta\mathrm{SCM}'$ both contribute to these variations. For instance, in ImageNet, $\Delta\mathrm{Acc}'$ ranges from 0.037% to 1.570%, while $\Delta\mathrm{SCM}'$ 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 (Fig. 4a), while non-sequential DNNs have more complex branches, such as MobileNetV2 and ResNet50, as depicted in Figs. 4b and 4c. To characterize the fault impacts of these branch structures, max-policy is designed to handle them and verify their effectiveness with the comparative results in Fig. 8a, using ImageNet as an example.

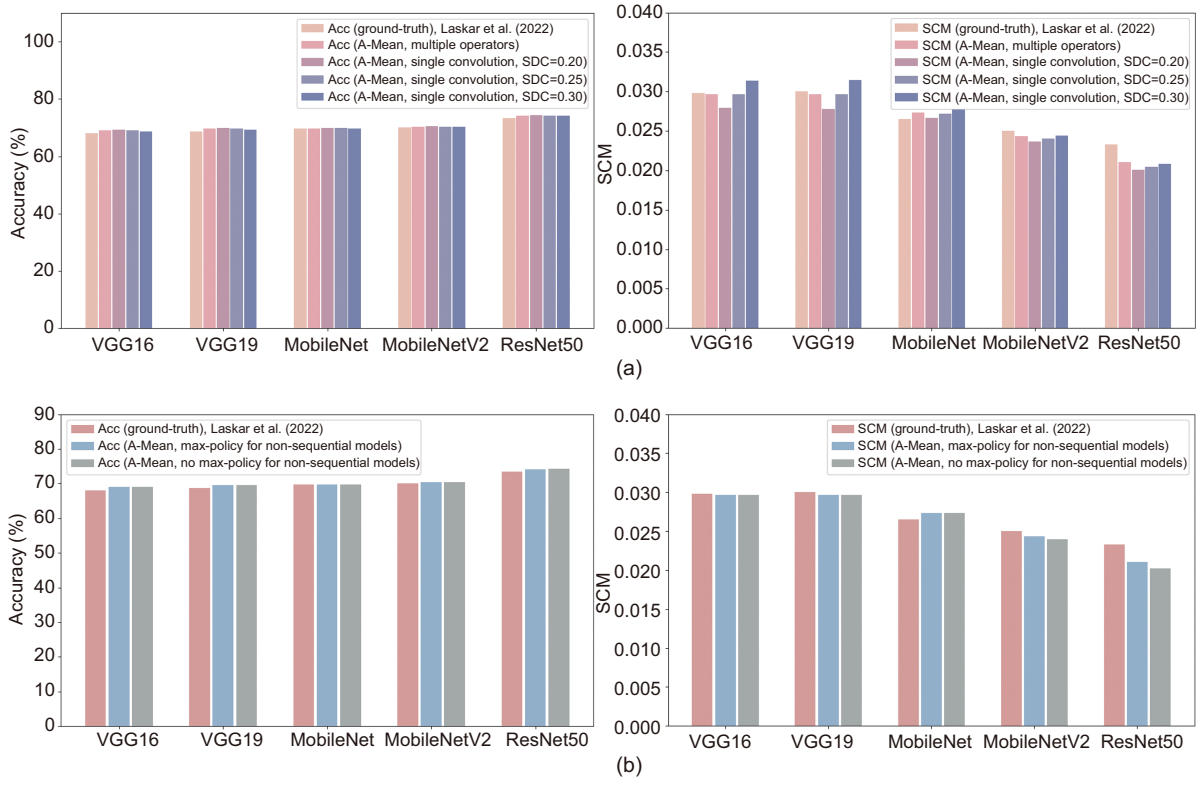

Fig. 8 SCM and accuracy improvement using max-policy on different strategies: (a) different topologies; (b) different operations

If the proposed A-Mean does not use the 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 accuracy of 70.4% and an SCM of 2.4%, while ResNet50 achieves an accuracy 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 accuracy of 70.3% and an SCM of 2.4%, and ResNet50 shows an accuracy 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 accuracy values are 70% (MobileNetV2) and 73.4% (ResNet50) and SCM values are 2.5% (MobileNetV2) and 2.3% (ResNet50), 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 is more accurate in classifying data 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 Figs. 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 branches in Fig. 4b, both parts contain various hidden layers. Max-policy can effectively integrate features from both parts of Figs. 4b and 4c, resulting in favorable outcomes.

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

Convention operations hold a dominant position in DNNs, and fault occurrence in the related components certainly affects the accuracy and SCM of the DNNs. If our proposed method A-Mean considers only convention operations, there will be a large gap between A-Mean and the ground truth (Laskar et al., 2022), as Fig. 8b shows, using ImageNet as a case study.

Three experiments are conducted on A-Mean with a single convention operation, with $\mathrm{SDC}_{\mathrm{Conv}}$ set to 0.20, 0.25, and 0.30. The obtained accuracy of the three experiments ranges from 69.28% to 74.32%, 68.99% to 74.25%, and 68.71% to 74.18%, and SCM ranges from 2.01% to 2.79%, 2.04% to 2.96%, and 2.08% to 3.14%, respectively. Instead, if more operators such as Add, Mul, ReLU, and max-pooling are considered in the enhanced A-Mean method, the estimation accuracy ranges from 68.99% to 74.15% and the SCM value ranges from 2.10% to 2.97%. Compared to other $\mathrm{SDC}_{\mathrm{Conv}}$ values, when $\mathrm{SDC}_{\mathrm{Conv}}$ is set to 0.25, A-Mean with a single convention operation performs better. However, Mul operations perform the best among these comparisons.

An interesting phenomenon is that accuracy and SCM exhibit opposite trends for DNN models; higher accuracy and lower SCM are shown in Fig. 8. This inverse relationship results from our evaluation methodology using a unified metric SDC to assess both accuracy and SCM. Accuracy measures the extent of correct classification, while SCM measures the extent of misclassification. As a result, these 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 uses one-time dynamic execution to obtain the initial values under no fault conditions. It then employs a static two-level mean calculation mechanism to estimate the expansion rate for each layer, followed by calculating the final expansion rate SDC. Finally, A-Mean leverages the worst-case policy to quickly and accurately estimate the fault impacts on various metrics, such as accuracy and SCM.

However, A-Mean has the following limitations: (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 Conclusions

To accelerate the assessment of the transient fault impacts on DNNs, this paper presents a novel

static two-level mean calculation model, A-Mean. This proposed approach can take advantages of one-time dynamic execution and static analysis for accurate, fast, and automatic estimation using the worst-case policy. The static two-level mean calculation refers to inner-layer and inter-layer mean calculations. More importantly, the sequential and non-sequential DNN topologies are considered, 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 DNN's accuracy under hardware transient faults, with 23.18 to 922.80 times speedup, 0.13% to 1.42% accuracy loss and 0.09% to 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, processed the data, and drafted the paper. Hong YANG helped organize the paper. Jiajia JIAO and Ran WEN revised and finalized the paper.

Conflict of interest

All the authors declare that they have no conflict of interest.

Data availability

The artifact of A-Mean is publicly available at https://github.com/breatrice321/A-Mean. The other data that support the findings of this study are available from the corresponding author upon reasonable request.

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 DNNs' reliability assessment. IEEE European Test Symp, 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. *Microproc Microsyst*, 94:104648. https://doi.org/10.1016/j.micpro.2022.104648

Belčević NM, Stojanović ZN, 2022. Using voltage signals for transient fault detection on overhead lines. *Int J Electr Power Energy Syst*, 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. *Microelectron Reliab*, 76-77:68-74. https://doi.org/10.1016/j.microrel.2017.07.007

Chen ZT, Narayanan N, Fang B, et al., 2020. TensorFI: a flexible fault injection framework for TensorFlow applications. IEEE 31st Int Symp on Software Reliability Engineering, 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. IEEE/ACM Int Conf on Computer Aided Design, p.1-9. https://doi.org/10.1109/ICCAD57390.2023.10323809

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

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

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

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

Gavarini G, Ruospo A, Sanchez E, 2023. SCI-FI: a smart, accurate and unintrusive fault-injector for deep neural networks. IEEE European Test Symp, 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. 49th Annual IEEE/IFIP Int Conf on Dependable Systems and Networks, p.112-124. https://doi.org/10.1109/DSN.2019.00025

Jiao JJ, Marculescu D, Juan DC, et al., 2016. A two-level approximate model driven framework for characterizing multi-cell upsets impacts on processors. *Microelectron J*, 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. *J Syst Archit*, 130:102652. https://doi.org/10.1016/j.sysarc.2022.102652

Laskar S, Rahman H, Zhang BH, et al., 2022. Characterizing deep learning neural network failures between algorithmic inaccuracy and transient hardware faults. IEEE 27th Pacific Rim Int Symp on Dependable Computing, p.54-67. https://doi.org/10.1109/PRDC55274.2022.00020

Li GP, Hari SKS, Sullivan M, et al., 2017. Understanding error propagation in deep learning neural network (DNN) accelerators and applications. Proc Int Conf for High Performance Computing, Networking, Storage and Analysis, Article 8. https://doi.org/10.1145/3126908.3126964

Li PW, Zhen L, Li XJ, et al., 2021. Radiation hardness assurance of single event effects on components for space application. 4th Int Conf on Radiation Effects of Electronic Devices, p.1-6. https://doi.org/10.1109/ICREED52909.2021.9588712

Liang JH, Li YJ, Yin GD, et al., 2023. A MAS-based hierarchical architecture for the cooperation control of connected and automated vehicles. *IEEE Trans Veh Technol*, 72(2):1559-1573. https://doi.org/10.1109/TVT.2022.3211733

Mukherjee S, 2008. Architecture Design for Soft Errors. Morgan Kaufmann, Burlington, USA.

Papadimitriou G, Gizopoulos D, 2021. Demystifying the system vulnerability stack: transient fault effects across the layers. ACM/IEEE 48th Annual Int Symp on Computer Architecture, 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. IEEE 29th Int Symp on On-Line Testing and Robust System Design, p.1-7. https://doi.org/10.1109/IOLTS59296.2023.10224870

Ping LQ, Tan JWJ, Yan KG, 2020. SERN: modeling and analyzing the soft error reliability of convolutional neural networks. Proc Great Lakes Symp on VLSI, p.445-450. 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. 8th Int Conf on Reliability, Infocom Technologies and Optimization, 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. IEEE Symp Series on Computational Intelligence, 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. 25th Int Symp on Design and Diagnostics of Electronic Circuits and Systems, 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. 47th Annual IEEE/IFIP Int Conf on Dependable Systems and Networks, p.97-108. https://doi.org/10.1109/DSN.2017.30

Shi YD, Tong Z, Zhang ZX, et al., 2023. Research on jamming decision technology based on hierarchical architecture. 4th Int Conf on Electronic Communication and Artificial Intelligence, p.52-56. https://doi.org/10.1109/ICECAI58670.2023.10176793

Sun RH, Qiu PF, Lyu YQ, et al., 2021. Lightning: striking the secure isolation on GPU clouds with transient hardware faults. https://arxiv.org/abs/2112.03662

Taheri M, Ahmadilivani MH, Jenihhin M, et al., 2023. APPRAISER: DNN fault resilience analysis employing approximation errors. 26th Int Symp on Design and Diagnostics of Electronic Circuits and Systems, p.124-127. https://doi.org/10.1109/ddecs57882.2023.10139468

Tan JWJ, Ping LP, Wang QX, et al., 2023a. saca-AVF: a quantitative approach to analyze the architectural vulnerability factors of CNN accelerators. *IEEE Trans Comput*, 72(11):3042-3056. https://doi.org/10.1109/TC.2023.3283685

Tan JWJ, Wang QX, Yan KG, et al., 2023b. saca-FI: a microarchitecture-level fault injection framework for reliability analysis of systolic array based CNN accelerator. *Fut Gener Comput Syst*, 147:251-264. 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. Int Conf on Smart Generation Computing, Communication and Networking, p.1-9.

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

Zheng Y, Feng ZY, Hu Z, et al., 2021. MindFI: a fault injection tool for reliability assessment of MindSpore applicacions. IEEE Int Symp on Software Reliability Engineering Workshops, p.235-238. https://doi.org/10.1109/ISSREW53611.2021.00068

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