2022 IEEE Smartworld, Ubiquitous Intelligence & Computing, Scalable Computing & Communications, Digital Twin, Privacy Computing, Metaverse, Autonomous & Trusted Vehicles (SmartWorld/UIC/ScalCom/DigitalTwin/PriComp/Meta) | 979-8-3503-4655-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/SmartWorld-UIC-ATC-ScalCom-DigitalTwin-PriComp-Metaverse56740.2022.00111

Towards Robust Intelligence in Space

Xin Yuan, Ruolin Xing, Shangguang Wang, Mengwei Xu
Beijing University of Posts and Telecommunications

With ever-growing data amount generated on spacecrafts such as satellites, it becomes necessary to process part of those data in space with machine learning techniques before transmitting them to the ground. The key challenge for such in-space intelligence is its robustness due to the harsh environment those spacecrafts operate in, especially the single-event upset (SEU) that can cause the in-memory model weight to be flipped between 0 and 1. This work first builds a simulation platform that can efficiently and accurately inspect the robustness of Deep Neural Networks (DNNs) against SEUs. Atop the platform, we perform the first measurement study to demystify the DNN robustness against SEUs under both single-bit and multi-bit error settings. The results highlight how fragile DNNs could be in space, especially when the exponent bits within its weights are flipped. To this end, we propose an effective, model-transparent, and low-overhead approach to enhance the DNN robustness against SEUs. Its key idea is to offline scale up the weight of each layer to amortize the impacts from the flipped exponent bits, and then scale down the output with a scaling factor during execution. Extensive experiments show that our approach can reduce the DNN vulnerability by up to 50,000×, and thus make in-space intelligence robust enough even for critical tasks.

I. Introduction

Emerging low Earth orbit (LEO) satellite constellations generate massive amount of high-resolution Earth imagery. But satellite-terrestrial links are not capable of catching up, resulting in a large fraction of data left in space. On the other hand, images captured by satellites are often covered with clouds or uninhabited areas. Thus, even the data can be downloaded successfully. It's usually hard to extract much useful information. The straightforward solution is to process the onboard data and only download the critical or valuable portion to the ground. Led by the technology advancement in the aerospace and satellite industries, here comes the singular point that onboard computing power can support the in-orbit processing,i.e., *satellite computing* or even *"satellite-as-a-service"*. In-orbit processing typically incorporates deep learning (DL) techniques, covering a wide range of applications, such as remote sensing [1], [2], communication [3], [4], and spacecraft operation [5]–[7]. In the near future, we envision deep neural networks (DNN) powered AI will become a killer workload for space computing.

One of the fundamental requirements for in-orbit intelligent processing is *robustness*. Without the protection of the Earth's atmosphere, the onboard devices can be severely affected by the Sun's radiation. Single-event effect (SEE) is a typical radiation-induced damage. According to the investigation by NASA (1996) [8] and AEROSPACE (2009) [9] on the spacecraft anomalies, the SEE contributes to 38.7% and 46% of the total causes, respectively. SEE mainly has two forms of memory errors. The ones are transient Single Event Upsets (SEUs – "flipped bits"), and the others are potentially permanent failures like Single Event Latchups (SELs – "stuck bits"). SEUs may cause one or multiple bits in memory (SRAM) to flip between 0 and 1.

While a few bits seem trivial to the whole memory size (MBs or even GBs), flipped or stuck bits could cause significant disasters. For instance, the Russian spacecraft Phobos-Grunt crashed into the southern Pacific Ocean in 2012 because of a radiation-induced memory error [10]. SEEs are common in space: the latest European space-grade processor LEON GR740 is estimated to experience a staggering 9 SEUs a day on a geostationary Earth orbit [11].

Onboard machine learning (ML) based intelligent applications could also be affected by SEUs, leading to unacceptable damages. For instance, degraded model accuracy could cause the invaluable Earth imagery data to be ignored or the important disaster reports to be delayed. Therefore, it is of great importance to understand *to what extent SEUs could affect the performance of modern ML models and how we can possibly mitigate such negative impacts*.

To this end, we first built a simulation platform named SDLSim which inspects the robustness of DNNs against SEUs. SDLSim has a novel metric: the number of vulnerable SEU events that cause model accuracy degradation higher than a pre-defined error bar, e.g., 10%. This metric indicates once a SEU occurs, how likely the model will undergo a significant accuracy loss. SDLSim has two main advantages: (1) Comprehensiveness. SDLSim considers both single-bit flipping and multi-bit flipping. Such multi-bit memory errors are not randomly constructed but generated with physical patterns from prior work [12]. It also considers the memory layout of DNNs in given memory chips to ensure the integrity of testing. (2) Performance. The SDLSim design is based on the insight that flipped DNNs still have many identical layers. Therefore, a layer-wise caching mechanism can reduce a lot of redundant computations. As a result, it can achieve 5× faster processing than the vanilla implementation.

With SDLSim, we perform the first measurement study on the DNN robustness against SEUs under both single-bit and multi-bit error settings. Through testing on LeNet and MNIST, we make three key observations. First, very few bits being flipped by SEUs can cause significant accuracy degradation to DNNs. Even setting the error bar as high as

50% (which makes the model totally unusable), there are still 351,023 vulnerable events for those models, respectively. Second, multi-bit SEU events are much more harmful than 1-bit SEU events. Our experiments show that 2-bits SEU events are $7\times$ more likely to cause high accuracy degradation than 1-bit SEU events. This is because the model perturbation caused by each flipped bit could accumulate. Third, the DNNs are more sensitive to the exponent bits (especially its left bits) within the FP32 weights, as compared to the significant and fractional bits. Overall, the ratio of vulnerable SEU events within exponent bits is higher than 10%, while the significant bits and fractional bits are all less than 1%. Within the 8 exponent bits, the left bits are more vulnerable to SEUs.

The above measurement results highlight the need for a robust mechanism to mitigate the impacts of SEUs in space. Traditional approaches mostly focus on the hardware aspects, i.e., radiation-tolerant or radiation hardened process. For instance, some chips use sapphire or gallium arsenide which is less susceptible to radiation than silicon-based ones. Some other chips leverage redundant hardware to ensure data integrity such as Triple Modular Redundancy (TMR) [13]. Those approaches sacrifice the hardware performance for reliability. Recent studies investigate software-level approaches to gain a better trade-off between the performance and reliability of in-orbit intelligence [14], [15]. These approaches often require significant modification to the DNNs or can only deal with 1-bit SEUs. Yet in practice, a SEU event can easily cause many bits to flip [12].

Is there any universal, model-transparent, and low-overhead approach to effectively protect DNNs from SEUs? In this work, we give the positive answer with a novel DNN hardening technique named *layer-wise weights rescaling*. The key idea is to offline scale up the weight of each layer and scale down the output with a scaling factor during execution. Through such rescaling and restoring processes, the numerical impact of the exponent bits can be greatly amortized. The weights of the same layer often have close values. Thus, they can share the same scaling factor without accuracy degradation to reduce the storage and computation overhead of scaling factors. Furthermore, it's vital to avoid division overflow during the restoring process. We do not directly use the largest possible scaling factor, but add another indirect transformation during scaling.

We have evaluated the above layer-wise rescaling technique on 3 typical DNN models and 2 datasets. Compared to the vanilla DNN without revision, our approach can reduce the DNN vulnerability by up to 50,000$\times$. Our approach also outperforms prior work that makes lightweight modifications to the DNNs by more than 100$\times$. For further investigations, the evaluated system has been deployed on one of the satellites in Tiansuan Constellation [16]. Through the results collected on our ground stations for half a year, there is no false prediction being made by our in-space DNNs as compared to the original DNN executed on ground.

The major contributions of this work are as follows.

- We build SDLSim, an efficient and accurate simulator to inspect the DNN robustness in space. It can simulate multi-bit flipping scenarios for DNN weights.
- We perform a first of its kind measurement study to understand the DNN robustness under different SEU attacks. From the study, we gained crucial insights to design anti-SEU mechanisms at the software level.
- We propose a novel technique called layer-wise rescaling. It can effectively enhance the DNN robustness against SEUs, without compromising the model accuracy or runtime cost. Our experiments show that the DNN vulnerability can be reduced by three to four orders of magnitude by leveraging layer-wise rescaling.

II. Motivations and Related Work

A. Intelligence in Space

• **Space-native data** Satellites or international space stations (ISS) are generating massive data such as earth imagery, weather observations, and cosmic rays statistics [17]. Depending on certain missions, the data processing procedures may have to be real-time, energy efficient, and secure. With onboard intelligence techniques, advanced data processing algorithms can be deployed to meet the above requirements. On the other hand, a large fraction of data can not all be downloaded due to the constraint of the satellite-terrestrial link and the high cost of ground stations rental. Instead, the data needs to be (pre-)processed and only critical or more compact results are to be transmitted to ground. For example, small NN models can be deployed on satellites to filter out the images containing no objects of interest. Only those valuable earth images will be sent to ground for further analysis [1].

• **High availability** Nowadays, the majority of the Earth and billions of people still have no access to the Internet. While satellite constellations like Starlink promise to deliver network to those areas, certain applications cannot tolerate a long network routing to a remote cloud datacenter [18]. For example, the computing in orbit could benefit a multi-user gaming application. In-space datacenters could be an essential extension of edge computing and "near-data processing" paradigm.

• **Green space service** Satellites are born with zero-carbon nature. Compared to the solar energy harvested on ground, satellites have a much higher solar-to-electricity converting ratio due to more direct exposure to the sun. With new microwave technology, the power conversion efficiency can be even pushed to over 80% [19]. As the LEO constellation rapidly expands, the green computing power is scaling accordingly by leveraging the embedded computing units onboard [20]. Thus, LEO constellations have huge potential to provide greener services compared with the terrestrial datacenters.

• **Onboard AI applications** In-orbit ML has already been initiated. Equipped with onboard AI capability, PhiSat-1 is the first ever satellite launched to explore the efficiency of transmitting the Earth observation data [21]. In our preliminary prototype in "Tiansuan Constellation" [16], we have deployed an object detection model on a LEO satellite (Baoyun). It promises to reduce the space-ground network traffic by more than 90% as compared to all streaming to ground. In the future, we envision DNN-powered AI will become a killer workload for space computing.

B. Reliability in Space

As a complex radiation environment, space is full of high-energy particles that produce a variety of environmental effects on spacecraft like satellites and result in spacecraft anomalies. Among many forms of threats, Single Event Effect (SEE) accounts for a large proportion. According to the investigation by NASA (1996) [8] and AEROSPACE (2009) [9] on the spacecraft anomalies induced by the space environment, the SEE contributes to 38.7% and 46% of the total causes, respectively. SEE can be further divided into different error types such as Single Event Latchup (SEL) and Single Event Upsets (SEUs). Among them, SEU is the most common one: as a soft and non-destructive error, it behaves as transient pulses in logic or support circuitry, or as bitflips in memory cells or registers. Hard errors like SEL cannot be fixed at software level, therefore this work specifically targets at SEU in memory under the context of in-space AI applications. Prior work has shown that even a one-bit flip in memory can cause serious software malfunction [14].

There have been extensive efforts to mitigate the potential damage from SEU. Hardware hardening [22], [23] is a common approach to physically protect the processors from the high-energy particles. This approach, however, can only filter out only partial harmful particles and is orthogonal to the software-level method presented in this work. Redundancy is another option: error correction code (ECC) memory can automatically fix 1-bit errors [24]; or, a task can be dispatched to many processors and aggregates the output from them through a majority voter. Those approaches sacrifice hardware performance and flexibility.

There have been very few studies investigating the SEU for AI applications specifically. [14] exploits the DNN features in triple modular redundancy ECC techniques, to trade off the protection overhead and design robustness. [15] further introduces a novel median feature selection technique to filter the impact of bit errors prior to the execution of each layer, as a fine-grained extension to modular redundancy scheme. Those approaches can mostly deal with only 1-bit SEUs, while in practice one SEU event can cause many bits to flip.

C. Bit-flip DNN Attacks and Defense

As a type of adversarial attacks on DNNs, bit-flip attacks [25]–[27] seek to flip a small number of parameters in DNNs to destroy the model usability. At software level, there have been a few anti-SEU mechanisms proposed. For instance, [27] achieves this through averaging over a grain of weights followed by an appropriate combination of quantization and clipping of weight values in a grain. Those approaches often require non-trivial modifications to the DNNs and thus lead to accuracy degradation. Moreover, they mostly target specific threat models, e.g., gradient-based bit-flip attacks, instead of random SEU events.

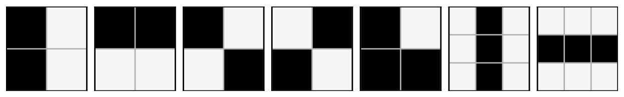


Fig. 1: Multi-bit SEU flipping patterns in memory [12] used in SDLSim. Black cells are flipped bits.

Fig. 2: SDLSim simplified workflow.

III. SDLSim: benchmarking and demystifying In-space Deep Learning Robustness

A. Metrics and Scope

All memory bits can be affected by SEU events. Therefore, it is important to reduce the common "vulnerable" bits in a DNN that can have severe impacts once being flipped. Meanwhile, a SEU event can influence multiple nearby bits and cause them to be flipped. Such multi-bit flip follows certain physical patterns. For instance, according to [12], there are 4 most common SEU-induced 2-bits error patterns, as shown in Figure 1.

We define a SEU event as vulnerable if it leads to the accuracy degradation of DNN higher than a pre-defined budget (named *errbar*), e.g., 10%. With this notion, the most concerned metric of this work is the number of possible vulnerable SEU events with given n-bits being flipped and an errbar. There are N 1-bit SEU events and $4 \times N$ 2-bits SEU events, respectively, where N is the total number of bits within the DNN. The accuracy before and after SEUs shall be tested on the same dataset.

This work focuses on the bits of model weights but not the feature maps generated during inference. This is because those intermediate feature maps are updated per input data, therefore they being flipped does not compromise the inference accuracy on other input. On the other hand, the model weights are stored in memory and used for each inference, therefore have a much higher sensitivity to the overall model performance.

B. Simulator Tool

The first contribution of this work is a simulation tool named SDLSim which can inspect the robustness of a DNN under SEU events in an accurate and efficient manner. Its

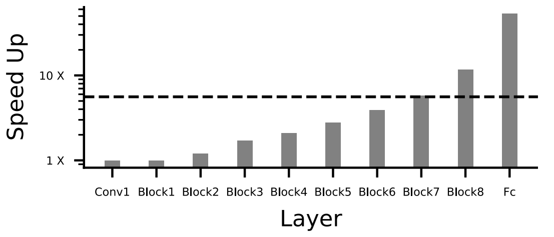


Fig. 3: The speedup achieved by SDLSim's caching mechanism on ResNet18's residual block.

input includes: the hardware specifications, especially the memory chip sizes; the DNNs; and the testing datasets.

A simplified workflow of SDLSim is illustrated in Figure 2. SDLSim first takes in the memory specification of the targeted hardware platform and generates flipped bit pairs on DNNs with its built-in multi-bit error patterns. For single-bit SEUs, the hardware does not affect the simulation. We only need to iterate over each bit of the DNN weights. SDLSim then iterates each possible flipped bit pairs of the DNNs according to their memory locations and generates a new DNN. For each flipped DNN, SDLSim tests the accuracy on the testing dataset and reports the accuracy drop compared to the original non-flipped DNN.

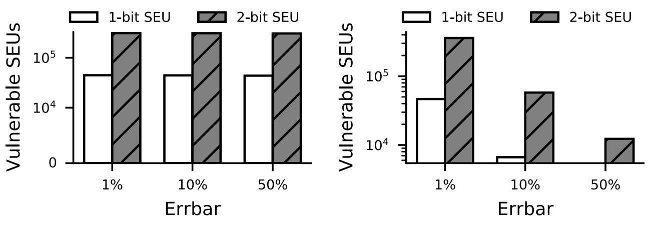
A key challenge in designing SDLSim is to improve the efficiency: given the huge amount of parameters of modern DNNs, the simulation process can be prohibitively slow. For instance, with the 4 possible 2-bits error patterns discussed above, the simulation computations take up to $4 \times 32 \times N \times D \times T_{infer}$, where 32 is the number of bits of FP32 format, N is the number of DNN weights, D is the testing dataset size, and T_{infer} is the inference complexity on one data sample.

To tackle the high computational complexity, we exploit the observation that *flipped DNNs often have shared layers*. That is, a SEU event could cause the 2 bits flipped at the last layer, therefore the computations of the first $N-1$ layers are identical. Based on the observation, we propose a layer-wise cache mechanism in SDLSim to avoid redundant computations and thus speed up the simulation process. More specifically, SDLSim always caches the intermediate feature map at each layer, and reuses it on the flipped DNNs that have identical prefix layers. As shown in Figure 3, such a cache mechanism can speed up the simulation process by 5–6×.

C. Preliminary Measurements

Based on SDLSim, we carry out a measurement study on how SEU events can affect the DNN robustness. Three key observations are summarized below.

Observation-1: very few bits can cause significant accuracy degradation to DNNs. Figure 4 illustrates the vulnerable SEU events (1-bit and 2-bits) for LeNet and MobileNet models. The X-axis represents different error bars (1%, 10%, and 50%). It shows that the number of vulnerable SEU events are huge on those models, i.e., 356,739 for LeNet and 405,830 for MobileNet with 1% error bar. Even setting the error bar as high as 50% (which makes the model totally unusable), there are still 351,023 and 12,280 vulnerable events for those models, respectively.



(a) LeNet and MNIST (b) MobileNet and CIFAR10

Fig. 4: Preliminary measurements on the amount of vulnerable event with different error bars.

Observation-2: Multi-bit SEU events are much more harmful than 1-bit SEU events. Figure 4 also compares the number of vulnerable SEU events under 1-bit and 2-bits flipping scenarios. It shows that 2-bits SEU events are much more likely to cause high accuracy degradation than 1-bit SEU events. This is because the model perturbation caused by each flipped bit could accumulate.

Observation-3: The DNNs are more sensitive to the exponent bits (especially its left bits) within the FP32 weights. DNN weights are mostly represented in 32-bit floating point (FP32) format, or more specifically, the IEEE Standard for Floating-Point Arithmetic (IEEE 754). Each FP32 number can be represented in the following form.

$$value = (-1)^{sign} \times 2^{(exp-127)} \times (1 + frac)$$

where $sign$ is the 1^{st} bit, exp is the $2^{nd} - -9^{th}$ bits, and $frac$ is the number represented by the last 23 bits in binary.

Table I breaks down the 1-bit vulnerable SEU events under 1% error bar into different layers and FP32 data components. A key finding is that the exponent bits are much more vulnerable than the significant and fractional bits. Overall, the ratio of vulnerable SEU events within exponent bits are higher than 10%, while the significant bits and fractional bits are all less than 1%.

We further break down the influence of SEU on the exponent bits at different positions (8 in total). Table II shows that the bits at higher (left) position are more vulnerable to SEUs. This is not surprising as the higher-positioned bits being flipped means the original weight being multiplied/divided by a larger number, i.e., 2^i for the i^{th} bit.

Implications The DNN robustness against SEUs needs to be hardened. The weakness of DNNs Explain: exp has a too high influence on the numbers: 2 exp XX. Much higher than others and can overwhelm the DL's original robustness.

IV. Design

Exponent bits analysis Since the exponent bits are the dominant source of DNN vulnerability, we focus on those bits to enhance the DNN robustness. For LeNet, the DNN weights are generally distributed between -1 and 1, and their absolute value is greater than 2^{-16}, therefore the exponent bits

Layer	Significant		Exponent.		Fractional	
	Total	Vul.	Total	Vul.	Total	Vul.
Conv1	156	4 (2.6%)	1248	165 (13.2%)	3588	0 (0%)
Conv2	2416	1 (0%)	19328	2588 (13.4%)	55568	0 (0%)
FC1	30840	0 (0%)	246720	30863 (12.5%)	709320	0 (0%)
FC2	10164	0 (0%)	81312	10164 (12.5%)	233772	0 (0%)
FC3	850	0 (0%)	6800	893 (13.1%)	19550	0 (0%)
Total	44426	5 (0%)	355408	44673 (12.6%)	1021798	0 (0%)

TABLE I: Vulnerable events (1-bit SEU and 1% error bar) categorized by the bit roles in FP32. Tested on LeNet and MNIST.

Layer	Total	Bit's Position in Exponent							
		1	2	3	4	5	6	7	8
Conv1	156	156 (100%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	2 (1.3%)	7 (4.5%)	0 (0%)
Conv2	2416	2416 (100%)	0 (0%)	0 (0%)	0 (0%)	11 (0.5%)	145 (6.0%)	16 (0.7%)	0 (0%)
FC1	30840	30840 (100%)	0 (0%)	0 (0%)	0 (0%)	14 (0%)	9 (0%)	0 (0%)	0 (0%)
FC2	10164	10164 (100%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)
FC3	850	850 (100%)	0 (0%)	0 (0%)	0 (0%)	1 (0%)	42 (4.9%)	0 (0%)	0 (0%)
Total	44426	44426 (100%)	0 (0%)	0 (0%)	0 (0%)	26 (0.1%)	198 (0.5%)	23 (0.1%)	0 (0%)

TABLE II: Vulnerable events (1-bit SEU and 1% error bar) for each-position exponent bits.

of most weights are in the form of "0111 xxxx". We break down the impacts of each single exponent bit in Table II.

- *The leftmost bit* The first bit has a very high impact on the model accuracy. In the case of LeNet, this bit has 100% chance to be vulnerable. This is because, when this bit is flipped from 0 to 1, the value of this weight will be multiplied by 2^{128}, which fully destroys the numeric balance among different weights.
- *The fixed bits* The 2^{nd} to the 4^{th} exponent bits are fixed to be 1 as discussed above. Being flipped from 1 to 0 means dividing the parameter by 2^k, which does not largely change the result of the layer. This is also confirmed in the table: those bits have 0% vulnerability.
- *The random bits* For the rest bits, assumed to be K^{th}, if it's flipped from 0 to 1, the parameter number will be multiplied by $2^{2^{(8-K)}}$, therefore the higher the bit in exponent bits, the larger the result of the layer where the weight is located will be affected. On the other hand, if the bit is flipped from 1 to 0, the number corresponding to the bit will be divided by $2^{2^{(8-K)}}$. As Table II shows, those bits have a chance to be vulnerable.

Note that the number of fixed bits varies across different DNNs as it relies on the model structure and training data.

Design rationales Based on the above analysis, we believe that in order to minimize the impact of SEU on the model accuracy, it is necessary to rescale the network model parameters, mainly by changing the exponent bits of the parameters. However, each parameter needs its own scaling factor, which further increases the vulnerability and runtime cost. To trade off the robustness and model accuracy, we leverage the fact that the weights within the same DNN layer are mostly concentralized across the same mean value. Therefore, we can adaptively choose a scaling factor per layer instead of per weight. It greatly reduces the extra cost to store those scaling factors while maintaining the robustness enhancement.

Our approach: layer-wise rescaling As shown in Algorithm 1, at offline, we first find the minimum value required to be added to the exponent bits of the weights in a layer to 0111 1111 as exp_mult. We then multiply the weights within this layer by 2^m, so that the model parameter exponent is as large as possible but does not exceed 0111 1111. Next, we shift the exponent bits of the layer parameter to the left by one, so that the highest bit 0 is erased. As shown in Algorithm 2, during execution, we add a new operator in the DNN to rescale the weights by dividing each weight by 2^m. In this way, most of the cases when the last 7 bits of the parameter exponent bits have a single-bit flip is to make the exponent of the model parameter smaller, and even if the parameter exponent becomes larger, it will not make the parameter larger too much, that is, it will not have a great impact on the model accuracy.

$$value = (-1)^{sign} \times 2^{(((exp+exp\_mult)<<1)-127)} \times (1+frac)$$

$$value = (-1)^{sign} \times 2^{((exp>>1)-exp\_mult-127)} \times (1+frac)$$

The first formula is to rescale the weights, and the second formula is to restore the weights.

A key challenge here is division *overflow*. If the parameter is directly multiplied by $2^{(128+m)}$ at offline rescaling, the first bit needs to be divided by $2^{(128+m)}$ during weights restoring as well. If the bit is 1, it will cause an overflow and huge model accuracy degradation. Therefore, we use an alternative scheme: after multiplying the parameters by 2^m, the last 7 bits of the exponent bits of the model parameters are shifted to the left by one bit, that is, the model parameters are as close as possible to 1111 1110. During weights restoring, the exponent bits of the model parameters are first shifted to the right by one and then divided by 2^m. In this case, the bit flip of the parameter exponent bits is equivalent to only occurring in the last 7 bits, and the accuracy loss from SEUs is greatly reduced by the transformation of the last 7 exponent bits.

V. Evaluation

Models and datasets Our experiments are performed on three classic DNN models: LeNet [28], ResNet-18 [29], and MobileNet [30]. These models are mostly small or medium-sized. Satellites are highly resource-constrained and therefore can hardly support large models like ResNet-152. Moreover,

Algorithm 1: Rescale Weights

```
Input: model's parameters:$parm$
Output: model's rescaled parameters:$new\_parm$;
        data mulitiplied to each tensor in
        $parm$:$exp\_mult$
Function Rescale():
    $exp\_mult$ = GetExpMult();
    $new\_parm$ = {};
    foreach $key$ in the $parm.keys$ do
        $new\_parm[key]$ = $parm[key]$ * $2^{exp\_mult[key]}$;
        foreach $weight$ in the $new\_parm[key]$ do
            $weight \longleftarrow$
              $weight's$ 8 bits exp left move 1 bit;
        end
    end
    return $new\_parm$,$exp\_mult$;
Function GetExpMult():
    $exp\_mult$ = {};
    foreach $key$ in the $parm.keys$ do
        $min\_mult$ = 127;
        foreach $weight$ in the $parm[key]$ do
            $min\_mult$ =
              min(127-$weight.exp$,$min\_mult$);
        end
        $exp\_mult[key]$ = $min\_mult$;
    end
    return $exp\_mult$;
```

Algorithm 2: Restore Weights

```
Input: model's rescaled parameters:$parm$; data
       mulitiplied to each tensor in $parm$:$exp\_mult$
Output: model's original parameters:$orig\_parm$
Function Restore():
    $orig\_parm$ = {};
    foreach $key$ in the $parm.keys$ do
        $orig\_parm[key]$ = $parm[key]$;
        foreach $weight$ in the $new\_parm[key]$ do
            $weight \longleftarrow$
              $weight's$ 8 bits exp right move 1 bit;
        end
        $orig\_parm[key]$ = $orig\_parm[key]$ / $2^{exp\_mult[key]}$;
    end
    return $new\_parm$,$exp\_mult$;
```

we choose DNN models for image processing, which is the key scenario of in-orbit computing. The datasets we used are MNIST [31] and CIFAR10 [32].

Baselines We compare our approach with two baselines: (1) `Vanilla-DNN`: the original DNNs without any revision; (2) `Activation-Replace` [33]: replacing the activation layers with more robust ones. We do not use redundancy-based approaches [13]. These approaches compromise on the execution cost (latency, energy, memory footprint, etc), and make the comparison unfair.

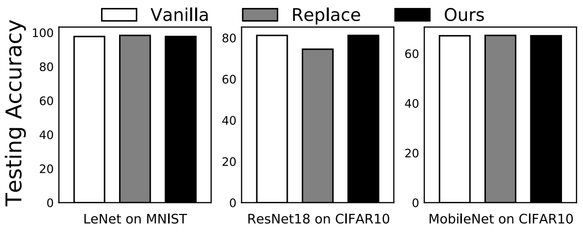


Fig. 5: The testing accuracy without SEU.

Setups and configurations By default, we assume the DNNs run on a memory chip of 256Kb×8. Each dataset is divided into the training set (80%) and the testing set (20%).

A. Simulation Results

Our technique has almost zero impact on the model accuracy. Figure 5 shows the testing accuracy of each approach. It shows the model accuracy is not affected after applying our technique. This is because our layer-wise rescaling guarantees the model equivalence to the original one. Instead, `Activation-Replace` changes the activation layers, e.g., from relu to tanh, which cause the models to be different. Consequently, it has somehow random impacts on the model accuracy, e.g., 6.6% degradation on ResNet-18 with CIFAR10.

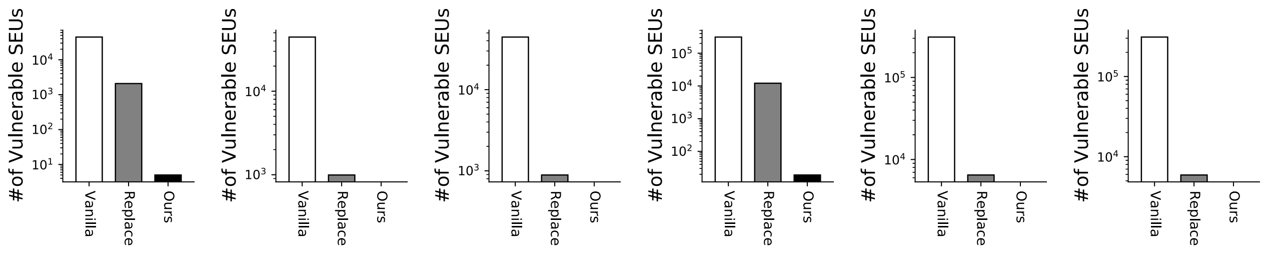
Note that the ability to ensure the model equivalence is crucial for the deployment of our approach. It means the technique can work in a fully automatic manner, and developers do not need to further tune the model for better accuracy.

Our technique significantly reduces the number of vulnerable SEUs and increases the model robustness. Figure 6/7/8 show the number of vulnerable SEUs for various models, error bars under 1-bit and 2-bits flipping. Generally, we observe that our technique significantly outperforms the baselines and can reduce the vulnerable SEUs to less than 10.

For 1-bit flipping on LeNet and MNIST, our approach reduces the number of vulnerable SEUs from more than 10,000 to less than 10, e.g., achieving up to 1,000× improvement. As comparision, `Activation-Replace` still encounters 2,000–3,000 vulnerable SEUs under different error bars.

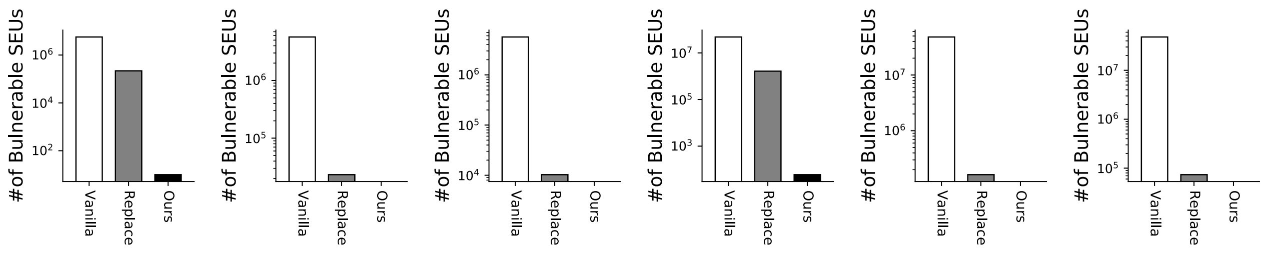
For 1-bit flipping on ResNet-18 and CIFAR10, our approach reduces the number of vulnerable SEUs from more than 1,000,000 to less than 20, e.g., achieving up to 50,000× improvement. Note that ResNet-18 is a much larger model than LeNet, therefore have more possible vulnerable SEUs (bits). As comparision, `Activation-Replace` approach still encounters 200,000–250,000 vulnerable SEUs under different error bars.

Under 1-bit flipping on MobileNet and CIFAR10, our approach reduces the number of vulnerable SEUs from 46,590 to 310 under 1% error bar and 6,650 to 0 under 10% error bar. The reduction is around 150×. The improvement of our approach is not as significant as on LeNet



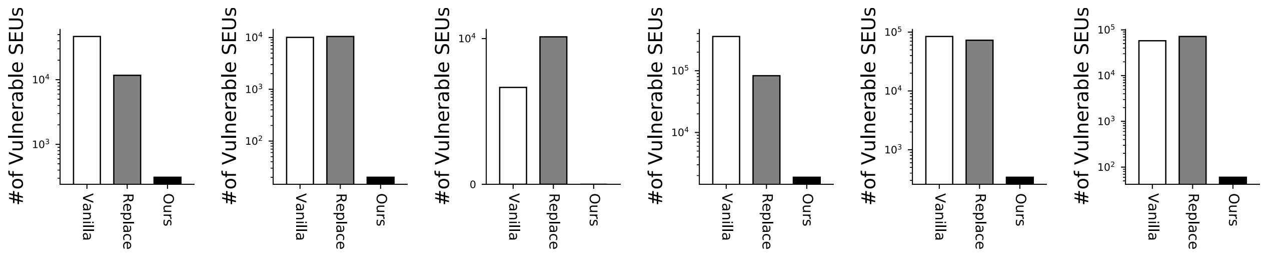
(a) (1-bit, 1%) (b) (1-bit, 5%) (c) (1-bit, 10%) (d) (2-bit, 1%) (e) (2-bit, 5%) (f) (2-bit, 10%)

Fig. 6: Vulnerable events on with LeNet and MNIST. Numbers in subcaption (x, y) mean the x-bit SEU and y error bar.



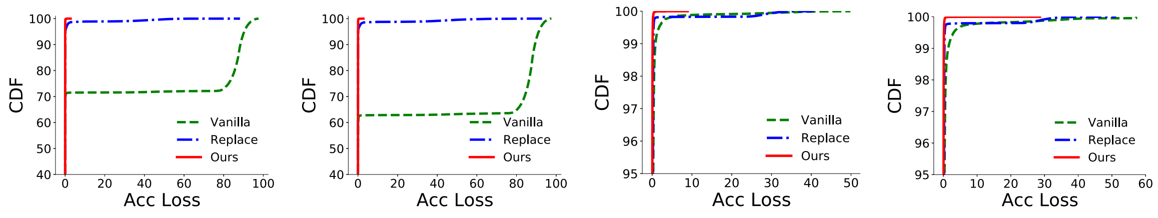
(a) (1-bit, 1%) (b) (1-bit, 5%) (c) (1-bit, 10%) (d) (2-bit, 1%) (e) (2-bit, 5%) (f) (2-bit, 10%)

Fig. 7: Vulnerable events on with ResNet18 and CIFAR10. Numbers in subcaption (x, y) mean the x-bit SEU and y error bar.



(a) (1-bit, 1%) (b) (1-bit, 5%) (c) (1-bit, 10%) (d) (2-bit, 1%) (e) (2-bit, 5%) (f) (2-bit, 10%)

Fig. 8: Vulnerable events on MobileNet and CIFAR10. Numbers in subcaption (x, y) mean the x-bit SEU and y error bar.



(a) LeNet, 1-bit SEU (b) LeNet, 2-bit SEU (c) MobileNet, 1-bit SEU (d) MobileNet, 2-bit SEU

Fig. 9: CDF of accuracy loss of SEUs on each possible bit(s).

and ResNet-18, because MobileNet is a more sophisticated structure and it is more sensitive to SEUs. Nevertheless, our approach still substantially outperforms the other baseline `Activation-Replace`, which has very limited improvement or even degrades the robustness under 10% error bar.

Figure 9 further illustrates the distribution of accuracy loss on each bit (or bit pairs for multi-bit SEUs). It shows that, on LeNet, our technique can bound the accuracy loss to less than 5% for each possible bit, while the other two baselines could result in more than 90% accuracy loss thus making the model totally unusable. On MobileNet, our approach experiences higher maximal accuracy loss (e.g., around 10% for 1-bit flipping and 30% for 2-bits flipping), but such circumstances occur very rarely. Meanwhile, the other two baselines experience even higher maximal accuracy loss, e.g., more than 50%. It shows that our approach not only reduces the possibility of significant-enough damages from SEUs, but also reduces the maximal damages from SEUs to the models.

B. *Sensitivity Analysis*

Our approach consistently outperforms baselines under both 1-bit and 2-bits flipping. Figure 6/7/8 also illustrate the performance on 2-bits flipping scenario. We observe that our approach can reduce the vulnerable SEUs by more than 1,000 as compared to the vanilla DNN and `Activation-Replace`. Note that our technique is the first to deal with multi-bit SEUs at the algorithm level, while prior approaches mostly focus on tackling single-bit SEUs.

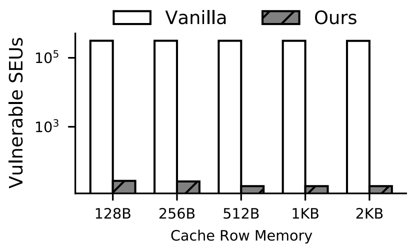


Fig. 10: Vulnerable SEUs under various row size of memory chips. Tested as multi-bits SEUs on LeNet and MNIST.

Our approach consistently outperforms baselines with different memory chips. We also evaluate our approach with different memory chip specifications, which could affect the bits flipped by different multi-bits SEU patterns. Figure 10 shows the DNN robustness under 5 memory chip row sizes. We observe that our approach has consistent enhancement on the DNN robustness.

VI. Conclusion

AI is becoming a promising workload for space computing. This work benchmarks, demystifies, and enhances the DL robustness against single-event upset (SEU). It first builds a benchmark tool to help developers inspect the DNN robustness, and gain insightful findings from a set of measurement studies. It then proposes a novel layer-wise rescaling technique to mitigate the bit-flipping impacts from SEUs, and thus makes DNNs more robust in harsh space environments.

References

[1] K. Li, G. Wan, G. Cheng, L. Meng, and J. Han, "Object detection in optical remote sensing images: A survey and a new benchmark," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 159, pp. 296–307, 2020.

[2] B. Denby and B. Lucia, "Orbital edge computing: Nanosatellite constellations as a new class of computer system," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2020, pp. 939–954.

[3] F. Fourati and M.-S. Alouini, "Artificial intelligence for satellite communication: A review," *Intelligent and Converged Networks*, vol. 2, no. 3, pp. 213–243, 2021.

[4] X. Wang, L. T. Yang, L. Kuang, X. Liu, Q. Zhang, and M. J. Deen, "A tensor-based big-data-driven routing recommendation approach for heterogeneous networks," *IEEE Network*, vol. 33, no. 1, pp. 64–69, 2019.

[5] P. V. R. Ferreira, R. Paffenroth, A. M. Wyglinski, T. M. Hackett, S. G. Bilen, R. C. Reinhart, and D. J. Mortensen, "Reinforcement learning for satellite communications: From leo to deep space operations," *IEEE Communications Magazine*, vol. 57, no. 5, pp. 70–75, 2019.

[6] F. d. G. Ortíz Gómez, R. Martínez Rodríguez-Osorio, M. A. Salas Natera, S. Landeros Ayala, D. Tarchi, and A. Vanelli Coralli, "On the use of neural networks for flexible payload management in vhts systems," 2019.

[7] X. Wang, L. T. Yang, L. Ren, Y. Wang, and M. J. Deen, "A tensor-based computing and optimization model for intelligent edge services," *IEEE Network*, vol. 36, no. 1, pp. 40–44, 2022.

[8] K. L. Bedingfield and R. D. Leach, *Spacecraft system failures and anomalies attributed to the natural space environment*. National Aeronautics and Space Administration, Marshall Space Flight Center, 1996, vol. 1390.

[9] J. E. Mazur, J. F. Fennell, J. L. Roeder, P. T. O'Brien, T. B. Guild, and J. J. Likar, "The timescale of surface-charging events," *IEEE Transactions on Plasma Science*, vol. 40, no. 2, pp. 237–245, 2011.

[10] "Phobos-grunt failure report released," https://www.planetary.org/articles/3361, 2012.

[11] "Space-grade cpus: How do you send more computing power into space?" https://arstechnica.com/science/2019/11/space-grade-cpus-how-do-you-send-more-computing-power-into-space/, 2019.

[12] S. Kumar, M. Cho, L. Everson, Q. Tang, P. Meinerzhagen, A. Malavasi, D. Lake, C. Tokunaga, M. Khellah, J. Tschanz, V. De, and C. H. Kim, "Analysis of neutron-induced multibit-upset clusters in a 14-nm flip-flop array," *IEEE Transactions on Nuclear Science*, vol. 66, no. 6, pp. 918–925, 2019.

[13] Y.-M. Hsu and E. E. Swartzlander, "Reliability estimation for time redundant error correcting adders and multipliers," in *IEEE International Workshop on Defect and Fault Tolerance in VLSI Systems*. IEEE, 1994, pp. 159–167.

[14] Z. Yan, Y. Shi, W. Liao, M. Hashimoto, X. Zhou, and C. Zhuo, "When single event upset meets deep neural networks: Observations, explorations, and remedies," in *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2020, pp. 163–168.

[15] E. Ozen and A. Orailoglu, "Boosting bit-error resilience of dnn accelerators through median feature selection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 3250–3262, 2020.

[16] S. Wang, Q. Li, M. Xu, X. Ma, A. Zhou, and Q. Sun, "Tiansuan constellation: An open research platform," *arXiv preprint arXiv:2112.14957*, 2021.

[17] M. Reichstein, G. Camps-Valls, B. Stevens, M. Jung, J. Denzler, N. Carvalhais *et al.*, "Deep learning and process understanding for data-driven earth system science," *Nature*, vol. 566, no. 7743, pp. 195–204, 2019.

[18] "SpaceX plans to start offering Starlink broadband services in 2020," https://spacenews.com/spacex-plans-to-start-offering-starlink-broadband-services-in-2020/, 2019.

[19] P. Malaviya, V. Sarvaiya, A. Shah, D. Thakkar, and M. Shah, "A comprehensive review on space solar power satellite: an idiosyncratic approach," *Environmental Science and Pollution Research*, pp. 1–17, 2022.

[20] D. Bhattacherjee, S. Kassing, M. Licciardello, and A. Singla, "In-orbit computing: An outlandish thought experiment?" in *Proceedings of the 19th ACM Workshop on Hot Topics in Networks*, 2020, pp. 197–204.

[21] "PhiSat-1," https://www.eoportal.org/satellite-missions/phisat-1, 2020.

[22] G. Lum, N. Bennett, and J. Lockhart, "System hardening approaches for a leo satellite with radiation tolerant parts," *IEEE Transactions on Nuclear Science*, vol. 44, no. 6, pp. 2026–2033, 1997.

[23] B. Singh, A. Foreman, and H. Trinkaus, "Radiation hardening revisited: role of intracascade clustering," *Journal of nuclear materials*, vol. 249, no. 2-3, pp. 103–115, 1997.

[24] C.-L. Chen and M. Hsiao, "Error-correcting codes for semiconductor memory applications: A state-of-the-art review," *IBM Journal of Research and development*, vol. 28, no. 2, pp. 124–134, 1984.

[25] A. S. Rakin, Z. He, and D. Fan, "Bit-flip attack: Crushing neural network with progressive bit search," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1211–1220.

[26] Z. He, A. S. Rakin, J. Li, C. Chakrabarti, and D. Fan, "Defending and harnessing the bit-flip based adversarial weight attack," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 14 095–14 103.

[27] J. Li, A. S. Rakin, Y. Xiong, L. Chang, Z. He, D. Fan, and C. Chakrabarti, "Defending bit-flip attack through dnn weight reconstruction," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020, pp. 1–6.

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

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[30] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[31] "The mnist database of handwritten digits," http://yann.lecun.com/exdb/mnist/.

[32] "The cifar-10 dataset," https://www.cs.toronto.edu/ kriz/cifar.html.

[33] W. Huiling, X. Zhuochen, and L. Xuwen, "Analysis and optimization of single event upset on neural network," *Journal of University of Chinese Academy of Sciences*, vol. 38, no. 6, p. 832, 2021.