Robust Machine Learning for Harsh Environments: A Framework and Evaluation of Key Architectural Choices

William Dennis[1 [0009−0005−4769−31362]] and James Pope[1 [0000−0003−2656−363X]]

University of Bristol, Intelligent Systems Laboratory, Bristol, United Kingdom

Abstract. Machine Learning algorithms are envisioned to be used in harsh and/or safety critical environments such as self-driving cars, aerospace, and nuclear sites where the effects of radiation can cause errors in electronics known as Single Event Effects (SEEs). The consequence of SEEs on machine learning models, such as neural networks composed of millions of parameters, is currently unknown. Understanding the models in terms of robustness and reliability is essential for their use in these environments. To facilitate this understanding, we propose a novel framework to simulate SEEs during model training and inference. Using the framework with different error models, we evaluate an exemplary Convolutional Neural Network (CNN) architecture and the Vision Transformer (ViT) architecture. For the CNN architecture we investigate robustness with varying dropout, regularisation, and activation functions. We then propose design decisions, specifically the choice of activation function and pooling approach, to improve the model's robustness. For the ViT architecture, we again investigate robustness using the framework. We find that the ViT is less robust than the CNN architectures considered with a higher drop in accuracy from the baseline (0.5840 vs 0.4042), we call this reduction the test delta. Overall, our results confirm the efficacy of the framework for evaluating model robustness in harsh environments.

Keywords: Robust Machine Learning · Fault Tolerance · Harsh Environments · Single Event Effects

1 Introduction

Deep learning architectures have shifted the paradigm from traditional, well-understood image processing methods towards less-understood models. The black-box nature of these state-of-the-art models presents concerns for critical applications that are exposed to harsh environmental effects. These are traditionally termed *Single Event Effects* (SEEs). The focus of this research is to develop and evaluate a framework for evaluating these complex models via simulated SEEs. SEEs are disruptions in electronic devices, often attributed to radiation, that could result in changes to the software running. Understanding how Machine Learning (ML) models behave under SEEs is vital for safety-critical applications, such as driverless systems.

Documented instances of electronic failure due to radiation include satellite communications [4], aircraft travelling at high altitudes [1], and computer games [34]. Robustness in ML systems are therefore a key requirement that has typically been referred to as the performance of a model under attacks to the input data. On the other hand, attacks to the parameter space in ML models are far less explored and do not fit under the adversarial robustness umbrella.

Convolutional Neural Networks (CNNs) are commonly used for image classification. AlexNet, a deep neural network (DNN), was introduced in 2012 and won the ImageNet Large Scale Visual Recognition Challenge [18]. AlexNet consisted of over 60 million parameters across 5 convolutional layers and 3 fully connected layers. These models are able to learn hierarchical representations of data with every layer abstracting features from the raw inputs. Subsequent models increased the number and complexity of the layers. More recently, Vision Transformers (denoted *ViTs*) have been used for similar image classification tasks with comparable performance [9]. ViTs divide the image into fixed-size patches, flatten them, and process them like a sequence of tokens using the Transformer encoder (originally designed for natural language processing tasks). These models also have a very large number of parameters. For example, the base ViT model has 88 million parameters across 12 encoding blocks [20]. DNNs and Transformers, with their many layers and huge parameter space, further exemplify the black box narrative.

There is limited research on SEE perturbations to the weights and biases of the CNN and ViT architectures. Research on SEEs is an extremely challenging area as the natural phenomena occurs at a very low probability, making experimentation very hard. Facilities, such as ChipIr can artificially generate SEE-like particles to perform beam experiments. This experimentation can produce more than 30 million years of natural irradiation in 241 hours [26]. Although this speeds up the natural process exponentially, experimentation is limited by the availability of these devices. Alternative approaches have looked towards fault injection software to simulate SEEs [11] but there is no standardised solution. These beam-experiments also focus on the sensitivity of hardware, and therefore suggest hardware solutions, with minimal regard to high-level radiation hardening techniques.

Our previous research [8], began to address this gap by exploring how CNN design choices influence the robustness to SEEs. In this research we summarise these results and add a comprehensive analysis of the ViT architecture. The contributions of the paper are as follows:

- Novel framework for evaluating neural networks in harsh environments[1].
- Propose two novel activation functions that are shown to be more robust to SEE related errors.
- Propose novel pooling layer that is also shown to be more robust to SEE related errors.

[1] Source code is available at https://github.com/wd7512/seu-injection-framework

2 Background

2.1 History of SEEs and Motivation

The idea of SEEs were first suggested [36] in 1962. In the 1970s, SEEs were known to disrupt spacecraft. Binder, Smith and Holman produced foundational work with their 1975 paper "Satellite Anomalies from Galactic Cosmic Rays" [4]. The authors detailed SEEs in operating satellites and explored the mechanisms by which cosmic rays impact microelectronics, resulting in satellite communication anomalies. Spacecraft are one example of SEEs in harsh environments. SEEs have also been shown to occur in everyday environments. A harmless example popularised by the media was a Mario speedrunning tournament where player was aided by a bit change (bitflip) in the memory of the computer running the game [34]. A more disastrous case was in 2008, during the flight of an Airbus A330 between Singapore and Perth. SEEs sadly resulted in 12 serious injuries when the A330 started nose-diving on two separate occasions [1,2]. Mitigation techniques are essential to enabling the use of electronics in these environments.

The miniaturisation of electronics exacerbates the problem. Transistors are the basic building block of electronics and have increasing sensitivity to SEEs as miniaturisation progresses. Moore's law states that the number of transistors on an integrated circuit doubles approximately every two years [21]. This has generally held true since the idea in 1965 with the modern day 3nm process transistors being less than 50nm in length [3]. The sensitivity arises from the critical charge required to change the state of the transistor being proportional to the size. Therefore as electronic devices become more powerful with more transistors, smaller transistors and higher transistor density, ionizing radiation requires less energy to induce an SEE. Consequently, the number of SEEs are expected to increase.

2.2 SEEs and Transistors

Edward Peterson defines SEEs as the action of a single ionizing particle as it penetrates sensitive nodes within electronic devices. SEEs in space arise from two natural sources, heavy ions from cosmic rays or solar flares and protons trapped in the earths magnetic field. An ion is an atom or molecule with a net positive or negative charge. This is a result of an imbalance between the number of electrons, that hold negative charge, and number of protons, that hold positive charge. Many devices can be upset by these ions at a rate of about 10^{-6} upsets per bit each day [22].

Named after Van Allen's discovery in 1958, the Van Allen radiation belts are zones of energetic charged particles originating from solar winds and held in place by the earths magnetic field [35]. There is an inner and outer region which can be dangerous for satellites to orbit. Peterson states that individual protons can be found here but have an energy level too low to upset most devices via direct ionisation. However, one proton in 10^5 will undergo a nuclear reaction within the body of a transistor capable of an upset. Therefore, at the heart of

the belt, 1KB of memory would have 10 upsets a day, a dangerously high number [22].

A heavy ion can deposit a track of ions as it passes through the body of the transistor shown in Figure 1. This produces an electric pulse or signal that may appear valid to the device, changing the state of the transistor. Protons passing through the body of the transistor have no immediate effect but can cause a nuclear reaction in the silicon. This can produce heavy ions that achieve the same effect.

Peterson divides SEEs into eight subcategories, focusing on Single Event Upsets (SEUs) and Multi-Bit Upsets (MBU). Both involve changes to computer memory via the change of a bit from 0 to 1 or 1 to 0. This is a type of Silent Data Corruption (SDC) and the focus of this research.

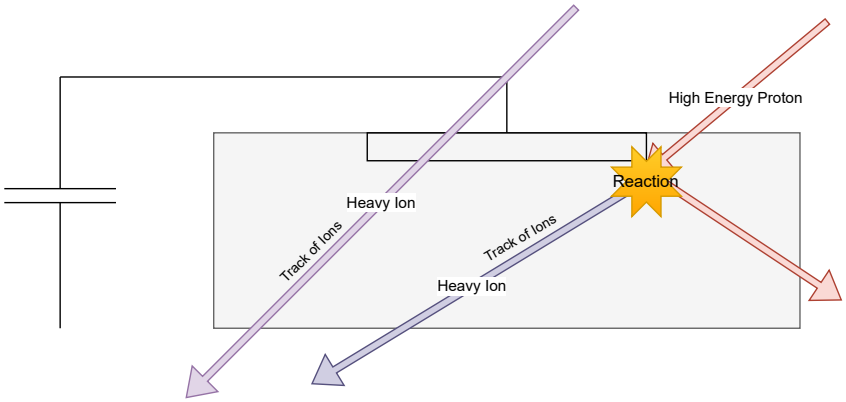

Fig. 1. Ionisation paths though a transistor that can disrupt the state (taken from [8]).

2.3 Numerical Representation in Computers

The IEEE 754 standard provides a framework for the representation of floating point numbers to varying degrees of precision [16]. This has been adopted as the industry standard and is consistent across all hardware. In ML, Floating Point (FP) 16, 32 and 64 are commonly used, where the number represents the number of bits used to represent the number. A higher number of bits allow for more precision, however the time taken for operations increases. Therefore, the choice of FP architecture is a trade off between speed and precision. The popular ML library sci-kit-learn uses FP64 by default, favouring precision. Deep learning frameworks PyTorch and Tensorflow-keras use FP32 by default as speed is more important [24]. The 32 bit standard represents numbers in the approximate range of $\pm 1.4 \times 10^{-45}$ to $\pm 3.4 \times 10^{38}$. Figure 2 depicts the architecture. The bits are

split into three parts that together constitute the floating point number. The first bit represents the sign of the number, bits 1 to 8 represent the exponent and the final 23 bits for the mantissa. The exponent is responsible for scaling the number by a power of two, while the mantissa determines the precision by representing the significant digits.

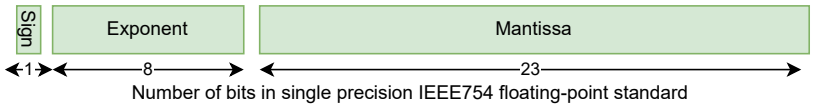

Fig. 2. IEEE 754 architecture for storing floating point numbers with 32 bits (taken from [8]).

It is expected that bitflips in the mantissa will have less impact than bitflips in the exponent or sign. This is because the sign bit can change the number by twice its magnitude and the exponent has the potential to send the number to the extremes of the FP32 range. Figure 3 shows the average result of performing bitflips on one thousand floating point numbers in the range of -3 to 3. Indeed, the Mean Absolute Error (MAE) of the first bit in the exponent has the largest impact. It is also interesting to note that the first several exponent bits (positions 1-6) result in more error than the sign bit (position 0).

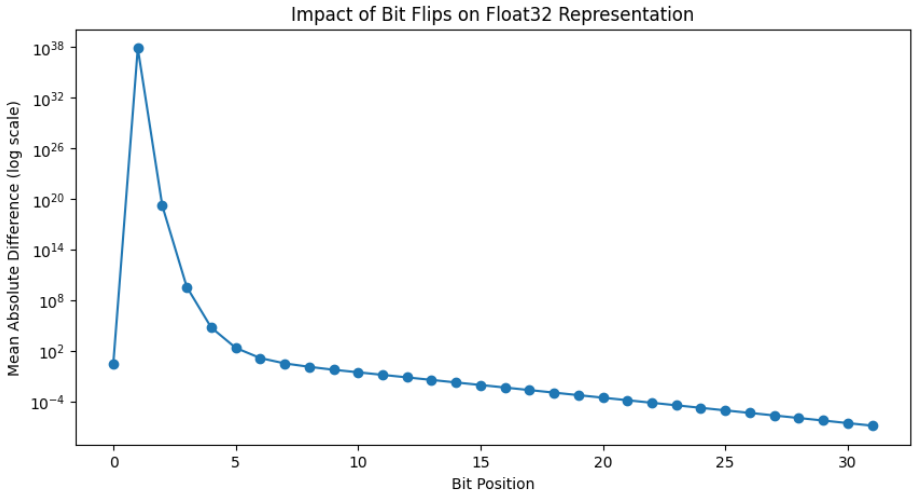

Fig. 3. Mean Absolute Error between FP32 numbers before and after performing a bitflip across each index. (taken from [8])

2.4 Convolutional Neural Networks

IBM defines a Neural Network (NN) as a machine learning program, or model, that makes decisions in a manner similar to the human brain by using processes that mimic the way biological neurons work together to identify phenomena, weigh options and arrive at conclusions [14]. The models consist of simple building blocks where each layer has a number of artificial neurons and an activation function. NNs have become a popular choice for a wide range of tasks, outperforming many traditional models. The universal approximation theorem states that a neural network can approximate any continuous function to a desired degree of accuracy, given enough neurons and number of layers [12] [19]. This theory underpins the versatility and effectiveness of neural networks, allowing them to excel across various applications.

A single neuron is a function that receives a real value x as input, processes it, and produces a real value as output. Mathematically, a single neuron can be described by the Equation 1 where w is called the weight and b the bias.

NN layers are formed by stacking multiple neurons together. A linear layer, also known as a Fully Connected (FC) layer, is the simplest example. This allows for the input to be a vector, $x \in \mathbb{R}^a$ and the output is a vector of size r, where $W \in \mathbb{R}^{r \times a}$, and $B \in \mathbb{R}^r$, this function is expressed in Equation 2. If a NN is composed of a single layer, it is often referred to as a perceptron. If many layers are composed together it can be referred to as a Deep Neural Network (DNN).

$$f(x) = wx + b \qquad x \in \mathbb{R} \tag{1}$$

$$f(x) = Wx + B \qquad x \in \mathbb{R}^a \tag{2}$$

The weight W and bias B parameters are learnt through the minimisation of a loss function that evaluates the NNs performance on a particular task. Depending on the task, this could be mean squared error or cross-entropy loss which are popular for regression and classification tasks respectively. Gradient descent is a common algorithm for minimising the loss and updates W and B. This can sometimes result in overfitting. Regularization is a technique used to mitigate overfitting by adding an additional term to the loss function that penalises large weights. This is also termed *weight-decay*. This encourages the model to generalise better. The linear layer, described in Equation (2), is inherently limited in its capacity to model complex, non-linear relationships within the data. This limitation is addressed by the introduction of activation functions. Activation functions add non-linearity to the NN.

CNNs have been designed and used to address problems in the computer vision field. They are a DNN composed of a feature engineering block and classification block. A CNN introduces a new type of layer called the convolutional layer modelled off the human eye. A convolution is a mathematical operation that slides one function over another. In the context of images, a kernel (filter) is applied across the image to generate feature maps. Pooling layers are another specialised layer that are paired with convolutional layers. The pooling layer

aims to reduce the spatial dimension of the feature maps, this process can be thought of as downsampling a signal. This reduces computational complexity and aids the spatial invariance by extracting the most important features. The most common type of a Pooling Layer is MaxPooling which takes the largest value in a kernel. The feature engineering block is made up of convolutional layers in conjunction with pooling layers and activation functions.

A three-dimensional array of features is the output of the feature engineering block. This is flattened and passed to the classification block via a dropout layer. During training, a dropout layer sets a number of the features to 0 with probability p. The authors of AlexNet state that this technique reduces complex co-adaptations of neurons, since a neuron cannot rely on the presence of particular other neurons. This encourages the model to learn more robust features. The classification block uses one or more FC layers with activation functions to convert the features into an output. In the final layer, a SoftMax function is used to convert the output into probabilities of each class.

3 Related Work

In the 21st century, research on SEEs has been directed towards the testing of AI hardware accelerators such as graphical processing units (GPUs). Paolo Rech provides a comprehensive review "Artificial Neural Networks for Space and Safety-Critical Applications: Reliability Issues and Potential Solutions" summarising the challenges and potential mitigation strategies for deploying NNs in environments where SEEs are a significant concern [25]. Rech states that a drawback of this parallelism in GPUs is that faults can propagate to multiple values. Therefore the resulting effect can lead to errors that impede the safe deployment of DNNs [33] [15].

How SEE related errors propagate through a DNN is currently not well understood. Matrix multiplication is the dominant process within DNNs and is what AI hardware accelerators are often used to compute. Therefore to assess the reliability of AI accelerators and DNN models under the effects of SEEs, beam experiments are used. Beam experiments are performed at facilities such as ChipIr at the Rutherford Appleton Laboratory, UK, [30] and have the capacity to irradiate microelectronics with atmospheric-like neutrons.

A detailed study across multiple GPUs was performed by Santos, et al., at ChipIr [29]. They show that a single fault tends to propagate to multiple threads, significantly reducing the reliability of CNNs . Errors with the rectangle distribution of affected parameters in a tensor were the most damaging to CNNs, and found to be caused by faults in the scheduling or the execution of multiple threads. This is intuitive as the rectangle shape will typically impact a larger proportion of parameters in a tensor than the single or line distribution. Santos, et al. [28], extended this work with *DieHardNet* that focuses on smart pooling and evaluates with residual deep neural networks and the CIFAR dataset. Our work extends and improves on this by considering and proposing activation

functions and evaluating with recent vision transformers and the more complex ImageNet dataset [27].

Error Correcting Codes (ECCs) reduced the total number of SDCs, but were unable to reduce the number of critical errors, which have the potential to impact safety-critical applications. The ECCs suggested were Algorithm Based Fault Tolerance (ABFT) and a redesign of the pooling layers. ABFT is a method where the final row and column of a weight matrix is used to check the integrity of the weights [13]. However Santos states AFBT is unable to be implemented on DNNs using NVIDIA proprietary cuDNN libraries. ABFT was however tested with FFT algorithms on GPUs and shown to be very promising [23] and where it could be tested at ChipIr, corrected 60% of the SDCs on the Tesla K40 and 50% on the Titan X [29].

A separate beamline experiment on Google's Tensor Processing Units (TPUs) found critical error rates of up to 4.73% during image processing tasks [26]. This study also explored the distribution of errors at the output of convolutional layers. This distribution of affected parameters falls in line with Santos's results. A drawback of the beamline experiments are the limited number of testing facilities and inability to run multiple experiments at the same time. For example Junior et al. had to test the TPU for more than 241 effective hours [26] (without considering the setup, load input, download output, and reboot time).

To support analysis from beamline experiments, artificial fault injection algorithms are also used. SASSIFI is an example of a hardware level algorithm that injects transient errors in NVIDIA GPU's [11]. Bosio et al. suggested the use of a high level fault injector, emphasising the independence of the fault injector from hardware, so that hardening techniques can be explored for both the DNN architecture and hardware [7]. Their analysis also showed that mitigation against critical behaviours was only required against changes to the exponent bits.

Introducing faults during the training process is an alternative strategy to overcoming hardware faults and shown to be create fault tolerance via the process of disabling random nodes during backpropogation [31]. This is an old ideas but there is little exploration into the robustness of models under this training procedure.

Practical research on SEEs has been shown to be both viable and valuable, however it is time consuming, costly and limited in terms of the extent of investigation. Exploring simulated methods can and should be explored as a complimentary and potentially advantageous alternative.

Our previous work [8] was the first to propose a framework for evaluating the robustness of neural networks for harsh and safety-critical environments, focusing on the CNN architecture. This work expands on this and adds a similar robustness analysis for the vision transformer, a more recent architecture for image classification.

4 Experimental Setup - CNN

4.1 CNN Use Case

Image classification for satellite imagery is chosen as a practical use case. Satellites exist in harsh environments with increased exposure to SEEs and commonly include imaging technology. The dataset used was called "Ships in Satellite Imagery" (also referred to as Shipsnet) [10]. Each image is made of three matrices of size 80 by 80. The three matrices represent the colours red, green and blue with values in the matrix between 0 and 255 for the intensity. The first 1000 images are part of the "ship" class with images centred on the body of a single ship. There are ships of different sizes, orientations and atmospheric conditions. The next 1000 are part of the "no-ship" class creating a binary classification problem. Figure 4 shows exemplary images from the dataset.

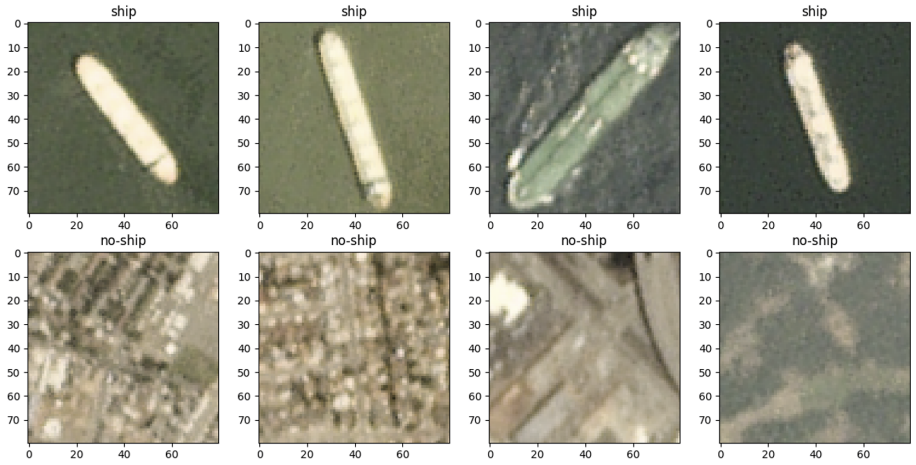

Fig. 4. Example Images from the Shipsnet Dataset (taken from [8]).

The CNN model for this research has three core requirements to meet. The model must excel at identifying images containing ships, be representative of larger, state-of-the-art models but also have a fast inference time to allow for efficient testing before and after SEE simulations. The number of layers, order, and layer size dictate the models complexity. Exploratory data analysis (EDA) on the dataset showed that the CNN architecture did not need to be a complex, state-of-the-art model with millions of parameters. However, to be representative of large scale DNNs the macro architecture of AlexNet was used [18]. This consists of a feature engineering block made up of convolutional layers, then a classification block made of fully connected layers. A dropout layer with probability p was inserted between these two blocks.

To train this CNN on the Shipsnet dataset, batched gradient descent is used on a random 80/20 train/test split of the 2000 samples. The loss function was

CrossEntropyLoss define in Equation 3 which measures the distance between the true labels and labels predicted by the CNN. Adam, a version of stochastic gradient descent, was chosen as the optimisation algorithm to minimise the loss.

$$\mathcal{L}(\mathbf{y}, \mathbf{p}) = -\sum_{i=1}^{C} y_i \log(p_i) \tag{3}$$

A batch size of 16 samples and a learning rate of 0.001 were chosen to balance training speed and model generalisation. The training is completed when either 1000 epochs are reached or the loss is less than 0.01. These values were selected as a loss of 0.01 indicates the predicted labels and true labels lie very closely. The 1000 epochs limit serves as a safeguard to stop algorithms that struggle to converge. Preliminary testing shows that all models converge within 100 epochs.

The hyperparameters explored in the training process were regularisation, also known as weight-decay, and dropout probability p. Regularisation adds an additional term to the loss function that represents the squared L2 norm of the model weights $\mathbf{w}$ multiplied by the regularisation coefficient λ seen in Equation (4). The aim of λ is to control the trade-off between fitting and overfitting by penalising large weights [32]. Values of λ at 0 and 1e-4 are chosen to compare the effectiveness of regularisation. The dropout probability p was explored in an off state, $p = 0$ and on state $p = 0.5$. Meaning that in the on state half of the 800 features are set to 0.

$$\mathcal{L}(\mathbf{y}, \mathbf{p}) = -\sum_{i=1}^{C} y_i \log(p_i) + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{4}$$

The hyperparameters explored in the CNN architecture were choice of activation function and the use of a smart pooling. The choice of activation function determines the non-linearity of the network. Table1 details the activation functions considered with their plots shown in Figure 5. The Weighted Gaussian activation function is denoted as WG and Rectified Weighted Gaussian, previously introduced in [8], as RWG .

Table 1. Activation Functions (taken from [8])

Function	**Expression**
ReLU	$\mathrm{ReLU}(x) = \max(0, x)$
Sigmoid	$\mathrm{Sigmoid}(x) = \frac{1}{1+\exp(-x)}$
Tanh	$\mathrm{Tanh}(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$
Sinusoidal	$\mathrm{Sin}(x) = \sin(x)$
Weighted Gaussian	$\mathrm{WG}(x) = x\exp(-x^2)$
Rectified WG	$\mathrm{RWG}(x) = \max(0, x\exp(-x^2))$

ReLU, Sigmoid and Tanh were chosen due to being well established across a range of NN models. In particular ReLU is a cornerstone of many CNNs such

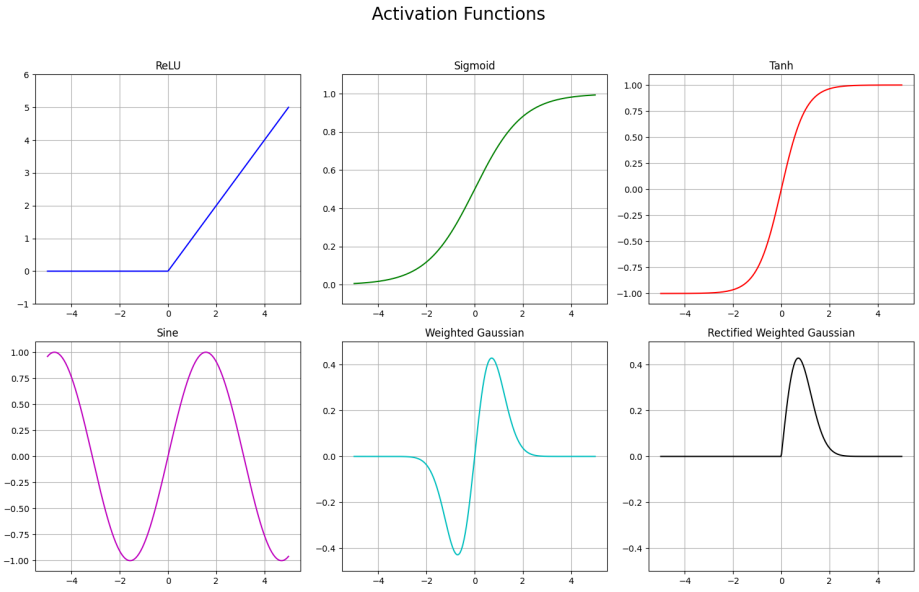

Fig. 5. Activation Functions Explored (taken from [8]).

as AlexNet [18]. We propose the following two activation functions specifically designed to handle SEE related errors, the Weighted Gaussian (WG) and Rectified Weighted Gaussian (RWG). The WG and RWG are bounded, decaying and zero-at-origin as these properties are hypothesised to aid robustness. This is because extreme changes will decay to 0. The sine function is included to assess the impact of periodicity, also being bounded.

Santos suggested the use of smartpooling and we explore this idea. [29]. This CNN replaces the MaxPooling layers with smartpooling layers that ignore unreasonable values [25]. Determining what is "unreasonable" can be done via many outlier detection techniques, the simplest being z-score or interquartile ranges. However due to the extreme magnitudes of resulting floats from bitflips, particularly at index 1 of the FP 32 architecture, an even simpler technique was implemented. All absolute values above a threshold were ignored. This aims to produce a similar result to more advanced outlier detection methods and serve as a proof of concept.

4.2 Error Model

With a use case and model in place, the final step was to create an environment to simulate and record the effects of SEEs. The aim is to bring the simulation as close as possible to empirical results, noting that there is very limited research in this area.

Figure 6 (taken from [8]) shows the design of the overall simulation process. The simulation takes a trained model as input and a test dataset for evaluation

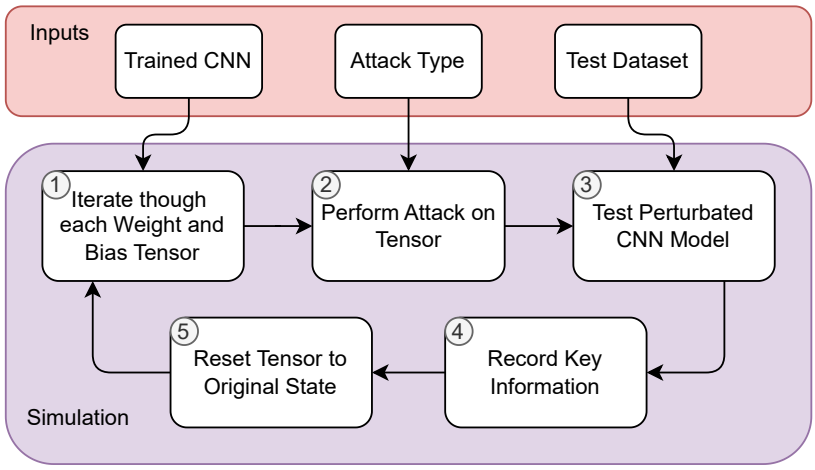

Fig. 6. A high level overview of the simulation environment (taken from [8]).

that is unseen data to the model. Then the model has its weights and biases erroneously modified (attacked) methodically. Each attack is self-contained in a weight or bias tensor and the tensor is reset to the original state after the attack. We do not consider the effect of cumulative SDCs. During the attack, the following key information is recorded: The **norm** defined in Equation 5, **name** of the layer, **location** of the attack, **accuracy** of the CNN on the test dataset, **number** of parameters attacked and **proportion** of parameters attacked in the tensor.

$$\text{norm}(t_0, t_1) = |t_0 - t_1| \tag{5}$$

To allow a fair comparison between different CNN model parametisations the bitflip index and spatial distribution must be consistent. Single, line and rectangle are common spatial distributions of SDCs. However, after preliminary exploration into simulations with these distributions, the line distribution was found to be catastrophic to the performance of the CNN. This allows for no comparison between parametisations so is unfortunately not a feasible simulation for a smaller CNN. This critical reduction in performance can be attributed to the low number of parameters in the CNN where feature channels are as small as 8. It will likely be more applicable to larger CNNs. The rectangle distribution is an even more destructive extension of the line distribution so was also not considered. This does not pose a problem as SEU is the most common result of SEEs.

Figure 6 depicts the simulation process used to analyse SEUs. The simulation comprehensively iterates through all 7634 parameters of the CNN. This approach was not possible for MBU simulations as the number of combinations is 2^{7634},

which exceeds practical computational resources. To represent an MBU, errors were propagated from an SEU to an MBU based on proximity of nearby bits using a Gaussian distribution. Spatial distributions of errors have previously been implemented by others [6]. The full details for simulating MBUs is presented in [8].

Figure 3 shows that the index of the bitflip can cause a near-zero change or change to the extremities of the FP32 architecture. For computational convenience, 7 of the 32 indices were selected to be investigated. The sign bit, three exponent bits (1,3,6) and three mantissa bits (10,15,21) were the indices selected. This gives a fair representation of the three parts of the FP32 architecture. The mantissa bits are skewed towards the lower indices due to the decreasing change in MAE. This is because it is expected that the impact would be minimal as the MAE approaches 0.

To summarise the simulation environment, both SEU and MBU methodologies were outlined. The SEU simulation exhaustively tests all parameters and the MBU attacks a selection of parameters propagating from a single SEU. Regarding the bitflipping, 7 of the total 32 bits in the FP32 architecture were tested. All this results in the following number of simulations:

SEU Simulations:

- Factors: 7,634 parameters, 6 activation functions, 7 bit indices, 2 pooling layers, 2 dropout, 2 regularization
- Total: $7,634 \times 6 \times 7 \times 2 \times 2 \times 2 = \mathbf{2,565,024}$

MBU Simulations:

- Factors: 458 initializations, 5 standard deviations, 6 activation functions, 7 bit indices, 2 pooling layers, 2 dropout, 2 regularization
- Total: $458 \times 5 \times 6 \times 7 \times 2 \times 2 \times 2 = \mathbf{769,440}$

5 Evaluation - CNN

To evaluate how the different model parameter settings affect the model's performance with different error models, we use the *test delta*. The test delta is computed as the difference between the test accuracy before and after the errors are introduced. Following our methodology, 3,334,464 attack simulations were conducted. Both SEU (2,565,024) and MBU (769,440) simulations were performed. Table 2 outlines the distribution of the number of simulations and the number of single bitflips. Note that the number of SEU attacks is greater than the number of SEU simulations. This is because a number of the MBU simulations, with a particularly low standard deviation, result in only one index being attacked.

5.1 Simulation

Table 3 offers a summary of the key simulation and CNN parameters against the test delta. It suggests complex interactions between these parameters, where one

Bitflips per Simulation	**SEU (1)**	**2-4**	**5-9**	**10-14**	**15-24**	**25-39**	**40+**
Total # Simulations	2,583,531	192,073	121,599	123,349	96,573	90,157	127,182

Table 2. Summary of Simulations Performed (taken from [8])

Dropout	**Weight Decay**	**Model**	**Standard Activation Functions**				**Proposed Act. Func.**	
			ReLU	**Sigmoid**	**Tanh**	**Sine**	**WG**	**RWG**
0.0	0.0	CNN	0.02760	0.01784	0.01758	0.01773	0.01347	0.01234*
		SmartPoolCNN	0.01809	0.01774	0.01756	0.01750	0.01363	0.01227*
	0.0001	CNN	0.02755	0.01731	0.01669	0.02002	0.01318	0.01275*
		SmartPoolCNN	0.02126	0.01738	0.01667	0.02023	0.01333	0.01260*
0.5	0.0	CNN	0.02418	0.01834	0.01573	0.01853	0.01322	**0.01020****
		SmartPoolCNN	0.02241	0.01828	0.01567	0.01857	0.01335	0.01021*
	0.0001	CNN	0.02155	0.01892	0.01702	0.01700	0.01314	0.01126*
		SmartPoolCNN	0.01931	0.01900	0.01718	0.01689	0.01313	0.01129*

Table 3. Average test delta of different CNNs and setups. The minimal value in each row is marked with an asterisk (*) and global minima with (**). (taken from [8])

parameter may only be beneficial in the presence of one or more other parameters. The analysis is visualised with multidimensional figures. The discussion starts with the simulation parameters, then considers the CNN's learning parameters and architectural design.

Bitflip location was the most important factor to the test delta. Figure 8 shows the non-zero test delta rate at each index. In the mantissa, only 1% of the simulations had a non-zero test delta, also known as critical error, with a maximum of only 0.0475 test delta. In the exponent, 27% of the simulations had a non-zero test delta with the 1st index having the most impact at 52%. It is surprising that the 3rd index had less impact than the 6th. These results support the hypothesis that the magnitude of the difference caused by the bitflip is proportional to the impact on the wider CNN.

Layer attacked within the CNN also had a large impact on robustness. Figure 8 shows a U-shaped pattern for average test delta with respect to the forward propagation order of the CNN layers. This points towards the first and final layers of the CNN architecture being the most sensitive to perturbations. The bias layers also exhibit higher test deltas than the weight layers suggesting they are more sensitive, a result that was also found by Bolchini et al. [5]. The most important takeaway from Figure 7 is that there is a clear linear relationship between the proportion of the layer attacked and test delta, confirming Santos' findings [29]. This is also reflected in Figure 8 where the SEU simulations have a lower test delta than the MBU simulations.

Dropout was a factor during the training of the models with the purpose of improving robustness. Averaged over all the results it reduced the average test delta from 0.0172 to 0.0164.

Regularisation was the least impactful parameter to the test delta having an insignificant increase to the test delta of 0.0001 over all the results. Regularisation only had a benefit when the ReLU activation function and dropout were used, shown in Table 3.

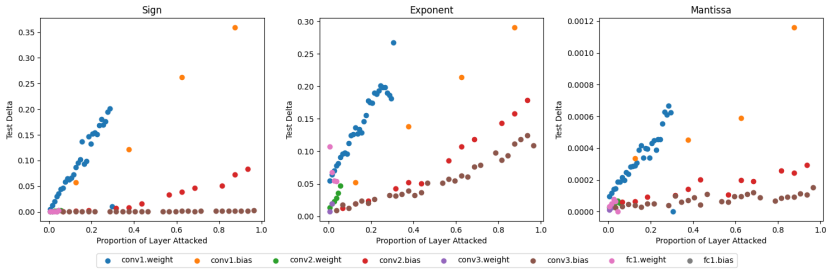

Fig. 7. The average test delta in the Sign (left), Exponent (middle), Mantissa (right) part of the FP32 architecture against the portion of the layer attacked (taken from [8])

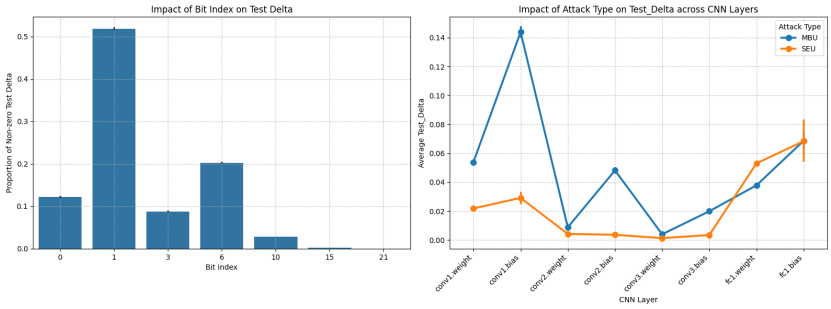

Fig. 8. Proportion of non-zero Test Deltas across Bit Indices (left), Average Test Delta across layers in the CNN (right). (taken from [8])

Activation Function had the most significant impact on the CNN architecture robustness. Moving left to right in Table 3, the activation function with the highest test delta was unbounded ReLU. Unbounded activation functions allow extreme values to propagate through the CNN. The bounded activation functions (sigmoid, tanh, sine) achieved similar levels of average test delta around 0.017. The bounded WG and RWG activation functions consistently achieved a lower test delta across all training parameters. The best model, at 0.01 average test delta, used the RWG activation function, dropout at $p = 0.5$, no regularisation and no smartpooling. This was a 58% reduction in error over the use of ReLU.

5.2 Testing AlexNet

The framework was then evaluated using AlexNet. It explored the RWG activation function and smartpooling. Due to computational constraints, transfer learning was used with the smaller CIFAR-100 dataset, which includes 60k images with 100 different labels [17]. The final layer of AlexNet was adjusted to the 100 classes instead of 1000. The images were scaled up to 224x224 pixels and normalised. After transfer learning, a pretrained version of AlexNet achieved a

Model Type	Test Accuracy	SEU Accuracy	Test Delta	# Simulations	# Training Epochs
Alexnet with ReLU	0.717100	0.312881	0.404219	160	20
Alexnet with $\text{RWG}_{0.01}$	0.742900	0.663311	0.079589	160	20* + 5
Alexnet with ReLU and smartpooling	0.707600	0.403594	0.304006	160	20*

Table 4. Comparison of Alexnet models on CIFAR-100. *Training epochs from weights copied from training Alexnet with ReLU. (taken from [8])

71.7% test accuracy as shown in Table 4. A variant of the RWG function (see [8]) allowed the use of transfer learning from AlexNet (which uses ReLU) to a version of AlexNet with the RWG_K activation function. Smartpooling had the advantage of not needing any further training, just the replacement of the pooling layers with smartpooling layers with minimal change in accuracy (decrease by 1%).

6 Experimental Setup - ViT

To further demonstrate the framework's flexibility we use it to evaluate the more recent vision transformer architecture [9] under SEUs. There are several ImageNet pre-trained vision transformers from the PyTorch library using different hyper-parameters [9]. We selected the *base* model using *32 patch size* as the most suitable with existing published results (denoted ViT_B_32). The ViT-Base architecture has 12 encoder blocks for a total of 152 learnable layers with a total of 88 million parameters, approximately 7 million parameters per encoder block.

We evaluate the model using weights pre-trained on the ImageNet-5K dataset [27], where the validation set has five images per class (with 1000 classes). With no errors being introduced, the vision transformer model achieves 76.08% accuracy on the validation set.

We consider SEU simulations and, as before, we select the sign bit, three exponent bits (1,3,6) and three mantissa bits (10,15,21) indices. For each index and for each layer, we randomly select 40 weights to attack. This results in a total of 6080 simulations per index. We note that the number of simulations are significantly less than previous experiments and is due to the fact that evaluating this much larger model takes significantly more time. The experimental setup is summarised as follows:

SEU Vision Transformer Simulations:

- Base Model: 152 layers (12 Encoding Blocks)
- Accuracy (no attacks): 76.08%
- Total simulations: 152 $layers \times 40$ $attacks\ per\ layer \times 7$ $bit\ indices = \mathbf{42,560}$

We also performed a few thousand more experimental simulations that altered individual tensor values to assess what magnitude is needed for the model prediction to differ.

Bit Index	Mean Accuracy	Mean Test Delta	Test Delta Std	Critical Failure %	# Simulations
0	0.7607	0.0001	0.0083	0.0000	6080
1	0.1768	0.5840	0.3071	0.7391	6080
3	0.7608	0.0000	0.0002	0.0000	6080
6	0.7601	0.0007	0.0203	0.0008	6080
10	0.7608	0.0000	0.0001	0.0000	6080
15	0.7608	0.0000	0.0000	0.0000	6080
21	0.7608	0.0000	0.0000	0.0000	6080

Table 5. Simulation results for SEU indices. Critical failure is defined as the proportion of samples with accuracy < 0.1.

7 Evaluation - ViT

Table 5 shows the results of the SEU simulation on the ViT. Again, we see that bit index 1 is the most damaging followed by index 6 and index 0. This aligns with Figure 8 from our CNN tests. Bit index 1 had a critical failure rate of 74% and a mean test delta (0.5840) that exceeds that of Alexnet (0.4042). This suggests that this ViT architecture is less resilient to SEUs than the Alexnet CNN architecture.

With two sets of experiments showing that the 6th bit index is more damaging than the 3rd, a deeper investigation was undertaken to understand this misalignment with Figure 3. It was found that the initial value before a bit occurs plays a large role in the resulting value. Specifically, values close to 0 have a greater change under bit 6 than bit 3. We know that many of the values in the CNN, Alexnet and ViT lie very close to 0 so this explains why bit 6 is more damaging than bit 3. Figure 9 shows this effect from performing bitflips on python float objects. Why python float objects behave this way is an unknown but it is unlikely that floating point values are sorted in this pythonic way for commercial use cases.

The layers with the highest average test delta are at the start of the ViT and after the transformer blocks. These are the convolutional layer that splits the image into smaller patches, the class token, then the post-transformer layer normalisations. Figure 10 shows how the mean test delta varies between the layers and Figure 11 explores within the transformer blocks.

The very first transformer block has the highest mean test delta whereas the following blocks maintain a similar level. We see a repeated pattern in Figure 11 which tells us that the same part of each block shares a similar sensitivity to SEUs. Table 6 shows the layer normalisation weight and bias have the highest test delta and critical failure rate. Therefore we see that throughout the ViT, both inside and outside the transformer blocks, layer normalisation is a sensitive part to alterations.

Unlike the CNN architecture the final fully connected layer (heads) in the ViT has a lower test delta than most of the model. This could also be due to the fact there are only 2 classes in the CNN tests and a 1000 for the ViT so a much lower proportion of this final layer is being altered. Given we are not doing

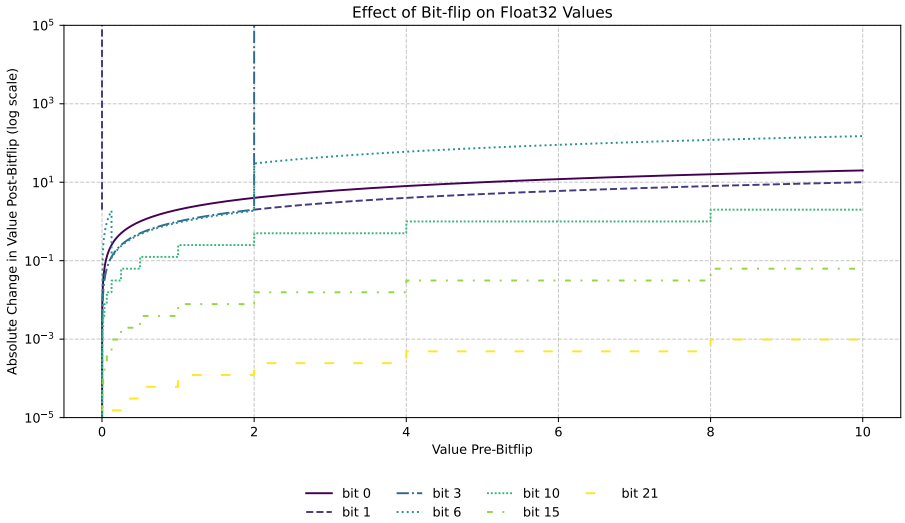

Fig. 9. Absolute Error between FP32 numbers before and after performing a bitflips at multiple bit indices across initial values between 0 and 10.

MBUs for the ViT experimentation, we must be mindful to compare the results from the ViT to only the SEU results of the CNN tests.

The positional embedding layer was extremely insensitive to SEUs with a 0% critical failure rate. The largest test delta was only 0.0094, meaning that only 47 images went from being classified correctly to being misclassified (5000 total with 3804 being classified correctly). A reason for this resilience could be due to the next step in the inference being a layer normalisation. This layer normalisation will take the values in each token plus its positional embedding and scale it. If an extreme value is present, the resulting output becomes a vector with all values set near 0 except the location of the extreme value, which now is a smaller, safer value. Therefore nearly all information is lost from this token/patch but this is just one of 50 tokens so most of the information remains intact. Consequently, the overall representation of the image is preserved, and the model can still make accurate predictions.

Matrix Name	Mean Accuracy	Mean Test Delta	Test Delta Std	Critical Failure %	# Samples
layer_norm.bias	0.6522	0.1086	0.2659	0.1429	6720
layer_norm.weight	0.6516	0.1092	0.2664	0.1436	6720
mlp.bias	0.7014	0.0594	0.2039	0.0781	6720
mlp.weight	0.6807	0.0801	0.2201	0.0835	6720
self_attention.bias	0.6897	0.0711	0.2207	0.0929	6720
self_attention.weight	0.6870	0.0738	0.2242	0.0961	6720

Table 6. Simulation results for parts of the transformer block. Critical failure is defined as the proportion of samples with accuracy < 0.1.

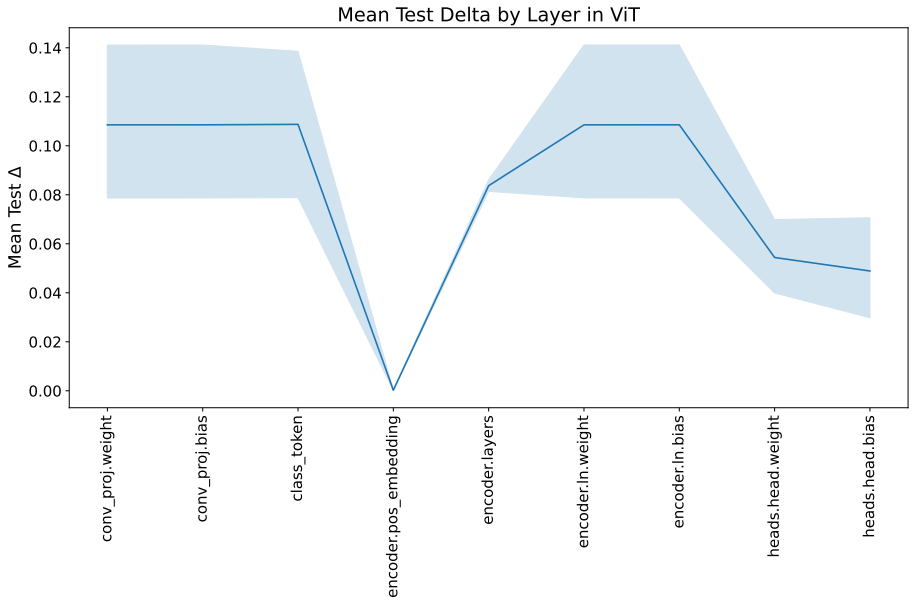

Fig. 10. Mean Test Delta for each layer in the ViT. Encoding layers are aggregated due to there being 12 transformer blocks

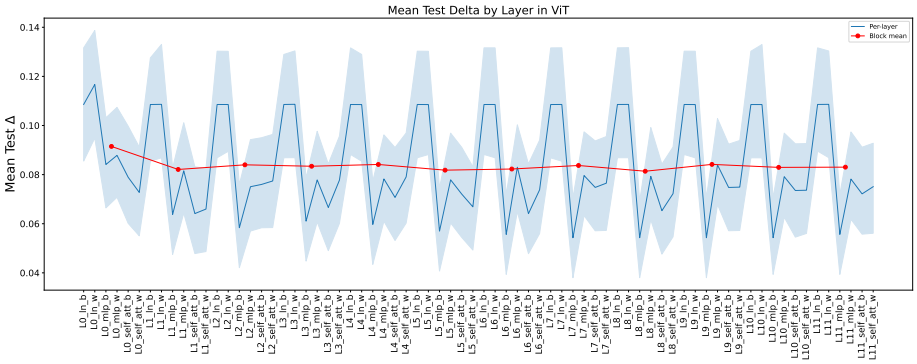

Fig. 11. Mean Test Delta for each layer in the ViT's twelve encoding blocks (six layers in each block).

Our CNN study showed a positive relationship between the log of the **Norm** and test delta. We also wanted to find out the minimum norm required to alter the accuracy of the ViT and the relationship between norm and performance of the ViT. Figure 12 shows that we only start seeing non-zero test delta at norm values above 0.1. At norm values greater than 10 we see our first critical failures which only get worse as we increase the norm. The smallest norm value to incur a non-zero test delta is found at 3.212×10^{-5}. The smallest norm value where we get more non-zero test deltas than zero test deltas is found at 1.314×10^{-1}. This value can be thought of as a tipping point where changes to a tensor greater than this will likely change the model's classification ability.

The ViT is found to handle norm values 10 times larger then the small CNN before a non-zero test delta is incurred. What this does not factor in is the increased size of the tensors in the ViT, thus a lower proportion of the tensor is altered. In the future, a cosine similarity score may be better than using the l1 norm as is incorporate dimensionality in the calculation. Regardless, this points towards the ViT being more robust to the magnitude of changes incurred by SEUs.

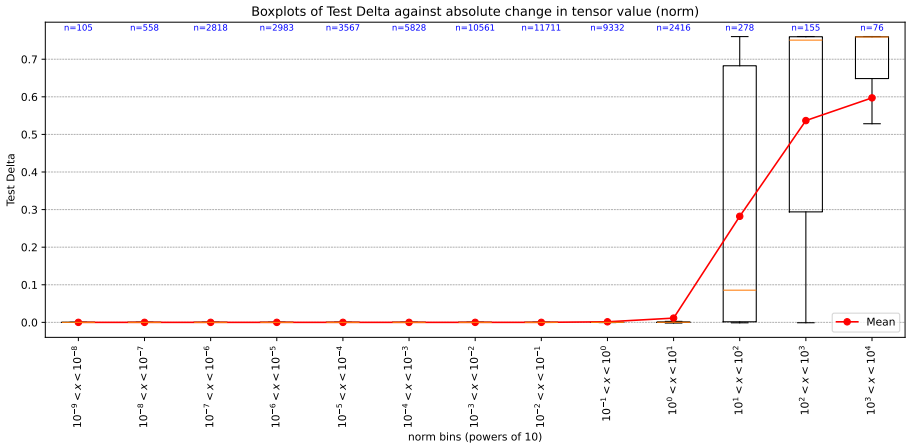

Fig. 12. Test Delta distribution across various binned values of norm (i.e. the absolute difference between the value before and after the SEU)

8 Conclusion

To conclude, a framework to simulate and test the effects of SEEs was built with the PyTorch library and successfully test three computer vision models, a small CNN, Alexnet and a Vision Transformer.

In the small CNN, a third of simulated SEEs have minimal impact due to small changes in the mantissa. There exists a linear relationship between the

proportion of the layer attacked and test delta. The use of dropout during training improves robustness but regularisation does not. Smartpooling layers were beneficial in conjunction with ReLU. The proposed activation function RWG was the most robust to simulated SEEs and transfer learning can be used to convert CNNs using ReLU to RWG_K. Small norms (e.g. less than 10^{-6}) have minimal impact on the test delta.

Both Alexnet and ViT have a parameter space in the tens of millions. Under an SEU which had a bitflip at the first exponent bit, Alexnet showed more resilience with a lower test delta than the ViT. This could be due to the convolutional layer skipping the altered tensor value. We also saw a huge improvement to the resilience of Alexnet when using the RWG activation function.

Within the ViT the positional embedding layer showed impressive resilience to SEUs. The normalisation layers in the ViT were the most least resilient across all 12 transformer blocks with the first transformer block being more sensitive than the rest. Similar to the small CNN, norms less than 10^{-6} have minimal impact on the test delta.

Future work will include testing more models and in more depth, similar to the thoroughness of the small CNN experimentation. This however does require compute proportional to the inference time of each model. Defining a bound on how much the parameter space can be altered would be an interesting analytical problem as our empirical results show this to be around 10^{-6} for the l1-norm. Cosine similarity could also be used for this. The mitigation techniques we suggested were not implemented for our ViT investigation so this is another obvious step forward. Alternative mitigation ideas to explore could be formulating a distribution for the values passed through the inferences, then rejecting values with a very low likelihood of belonging to this distribution. Finally, future work will examine real life results to better understand error distributions.

References

1. ATSB: In-flight upset 154 km west of learmonth. Australian Transport Safety Bureau (2008), https://www.atsb.gov.au/publications/investigation_reports/2008/aair/ao-2008-070
2. ATSB: Single event effect general risk to avionics. Australian Transport Safety Bureau (2011), https://www.atsb.gov.au/safety-issues/AO-2008-070-SI-07
3. Badaroglu, M.: More moore. In: 2021 IEEE International Roadmap for Devices and Systems Outbriefs. pp. 01–38 (2021). https://doi.org/10.1109/IRDS54852.2021.00010
4. Binder, D., Smith, E.C., Holman, A.B.: Satellite anomalies from galactic cosmic rays. IEEE Transactions on Nuclear Science **22**(6), 2675–2680 (1975). https://doi.org/10.1109/TNS.1975.4328188
5. Bolchini, C., Cassano, L., Miele, A., Nazzari, A.: Selective hardening of cnns based on layer vulnerability estimation. In: 2022 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT). pp. 1–6 (2022). https://doi.org/10.1109/DFT56152.2022.9962339

6. Bolchini, C., Cassano, L., Miele, A., Toschi, A.: Fast and accurate error simulation for cnns against soft errors. IEEE Transactions on Computers **72**(4), 984–997 (2023). https://doi.org/10.1109/TC.2022.3184274
7. Bosio, A., Bernardi, P., Ruospo, A., Sanchez, E.: A reliability analysis of a deep neural network. In: 2019 IEEE Latin American Test Symposium (LATS). pp. 1–6 (2019). https://doi.org/10.1109/LATW.2019.8704548
8. Dennis, W., Pope, J.: A framework for developing robust machine learning models in harsh environments: A review of cnn design choices. In: Proceedings of the 17th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART. pp. 322–333. INSTICC, SciTePress (2025). https://doi.org/10.5220/0013155000003890
9. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (2021), https://openreview.net/forum?id=YicbFdNTTy
10. Hammell, R.: Ships in satellite imagery (2018). https://doi.org/10.34740/KAGGLE/DSV/61115, https://www.kaggle.com/dsv/61115
11. Hari, S.K.S., Tsai, T., Stephenson, M., Keckler, S.W., Emer, J.: Sassifi: An architecture-level fault injection tool for gpu application resilience evaluation. In: 2017 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS). pp. 249–258 (2017). https://doi.org/10.1109/ISPASS.2017.7975296
12. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Networks **2**(5), 359–366 (1989). https://doi.org/https://doi.org/10.1016/0893-6080(89)90020-8, https://www.sciencedirect.com/science/article/pii/0893608089900208
13. Huang, K.H., Abraham, J.A.: Algorithm-based fault tolerance for matrix operations. IEEE Transactions on Computers **C-33**(6), 518–528 (1984). https://doi.org/10.1109/TC.1984.1676475
14. IBM: Neural networks. https://www.ibm.com/topics/neural-networks (2024), accessed: 2024-08-26
15. Ibrahim, Y., Wang, H., Liu, J., Wei, J., Chen, L., Rech, P., Adam, K., Guo, G.: Soft errors in dnn accelerators: A comprehensive review. Microelectronics Reliability **115**, 113969 (2020). https://doi.org/https://doi.org/10.1016/j.microrel.2020.113969, https://www.sciencedirect.com/science/article/pii/S0026271420308003
16. IEEE Standard for Floating-Point Arithmetic. IEEE Std 754-2019 (Revision of IEEE 754-2008) pp. 1–84 (2019). https://doi.org/10.1109/IEEESTD.2019.8766229
17. Krizhevsky, A.: Learning multiple layers of features from tiny images (2009), technical Report
18. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Pereira, F., Burges, C., Bottou, L., Weinberger, K. (eds.) Advances in Neural Information Processing Systems. vol. 25. Curran Associates, Inc. (2012), https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf
19. Lu, Z., Pu, H., Wang, F., Hu, Z., Wang, L.: The expressive power of neural networks: A view from the width (2017), https://arxiv.org/abs/1709.02540
20. maintainers, T., contributors: Torchvision: Pytorch's computer vision library. https://github.com/pytorch/vision (2016), accessed: 2025-06-28
21. Moore, G.: Cramming more components onto integrated circuits. Proceedings of the IEEE **86**(1), 82–85 (1998). https://doi.org/10.1109/JPROC.1998.658762

22. Peterson, E.: Introduction, chap. 1, pp. 1–12. John Wiley & Sons, Ltd (2011). https://doi.org/https://doi.org/10.1002/9781118084328.ch1, https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118084328.ch1
23. Pilla, L.L., Rech, P., Silvestri, F., Frost, C., Navaux, P.O.A., Reorda, M.S., Carro, L.: Software-based hardening strategies for neutron sensitive fft algorithms on gpus. IEEE Transactions on Nuclear Science **61**(4), 1874–1880 (2014). https://doi.org/10.1109/TNS.2014.2301768
24. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners (2019)
25. Rech, P.: Artificial neural networks for space and safety-critical applications: Reliability issues and potential solutions. IEEE Transactions on Nuclear Science **71**(4), 377–404 (2024). https://doi.org/10.1109/TNS.2024.3349956
26. Rech Junior, R.L., Malde, S., Cazzaniga, C., Kastriotou, M., Letiche, M., Frost, C., Rech, P.: High energy and thermal neutron sensitivity of google tensor processing units. IEEE Transactions on Nuclear Science **69**(3), 567–575 (2022). https://doi.org/10.1109/TNS.2022.3142092
27. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) **115**(3), 211–252 (2015). https://doi.org/10.1007/s11263-015-0816-y
28. Fernandes dos Santos, F., Cavagnero, N., Ciccone, M., Averta, G., Kritikakou, A., Sentieys, O., Rech, P., Tommasi, T.: Improving deep neural network reliability via transient-fault-aware design and training. IEEE Transactions on Emerging Topics in Computing pp. 1–12 (2025). https://doi.org/10.1109/TETC.2024.3520672
29. Santos, F.F.d., Pimenta, P.F., Lunardi, C., Draghetti, L., Carro, L., Kaeli, D., Rech, P.: Analyzing and increasing the reliability of convolutional neural networks on gpus. IEEE Transactions on Reliability **68**(2), 663–677 (2019). https://doi.org/10.1109/TR.2018.2878387
30. Science and Technology Facilities Council: Chipir (2024), https://www.isis.stfc.ac.uk/Pages/Chipir.aspx, [Accessed 01-10-2024]
31. Sequin, C., Clay, R.: Fault tolerance in artificial neural networks. In: 1990 IJCNN International Joint Conference on Neural Networks. pp. 703–708 vol.1 (1990). https://doi.org/10.1109/IJCNN.1990.137651
32. Shalev-Shwartz, S., Ben-David, S.: Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press (2014), chapter 13: Regularization and Stability
33. Su, F., Liu, C., Stratigopoulos, H.G.: Testability and dependability of ai hardware: Survey, trends, challenges, and perspectives. IEEE Design & Test **40**(2), 8–58 (2023). https://doi.org/10.1109/MDAT.2023.3241116
34. TheGamer: How an ionizing particle from outer space helped a super mario 64 speedrunner save time. https://www.thegamer.com/how-ionizing-particle-outer-space-helped-super-mario-64-speedrunner-save-time/ (16th Sep 2020), accessed: 2024-08-23
35. VAN ALLEN, J.A., LUDWIG, G.H., RAY, E.C., McILWAIN, C.E.: Observation of high intensity radiation by satellites 1958 alpha and gamma. Journal of Jet Propulsion **28**(9), 588–592 (1958). https://doi.org/10.2514/8.7396, https://doi.org/10.2514/8.7396
36. Wallmark, J.T., Marcus, S.M.: Minimum size and maximum packing density of nonredundant semiconductor devices. Proceedings of the IRE **50**(3), 286–298 (1962). https://doi.org/10.1109/JRPROC.1962.288321