The Impact of Proton-Induced Single Events on Image Classification in a Neuromorphic Computing Architecture

By

Rachel Mae Brewer

Thesis

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

in

Electrical Engineering

December 14, 2019

Nashville, Tennessee

Approved:

Brian D. Sierawski, Ph.D.

Ronald D. Schrimpf, Ph.D.

In memory of Grandaddy, for his encouragement, love, and support.

ACKNOWLEDGEMENTS

I would first like to thank my advisor, Dr. Brian Sierawski, for his guidance and support throughout this project. Additionally, I would like to thank Mike McCurdy for his help with the Pelletron and insights on single event testing. To Dr. Enxia Zhang, thank you for your help with delidding the TrueNorth chip. And to Drs. Ron Schrimpf, Michael Alles, and Robert Reed, thank you for your insight and feedback on this project. Finally, thanks to Steven Moran, Jon Cox, and Dr. Iyer at the University of California Los Angeles for their help throughout this project.

I would also like to thank my parents and grandparents for instilling in me a love of learning and encouraging me to pursue graduate studies. Thank you for your support and encouragement throughout graduate school.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Neuromorphic computing endeavors to imitate the way biological brains process information and problem-solve. Uses for neuromorphic computing span disciplines and include applications in image processing, audio processing, optimization, and more. Neural networks can be used for neuromorphic computing, and these neural networks can be implemented in software or emulated in hardware.

This work explores the effect of proton-induced single-event upsets on a neuromorphic computing architecture engaged in image recognition. First, background on neuromorphic computing, neural networks, and radiation effects is given. Next, the hardware and its operation are introduced followed by the setup used for the experiments. The remaining portion of the work covers the experimental results, simulation results, and the conclusions drawn from the experiments that are supported by simulation.

In the experiments, the overall classification accuracy is calculated after radiation and is compared to the accuracy before radiation. The changes in output classification is also explored along with the effects of tolerable and critical errors produced by single event effects. In addition, the effect of radiation on the classification of individual digits is analyzed as well.

Two main results are found. One, the overall classification accuracy is unchanged although a high number of hidden, tolerable errors occurred. Additionally, single-event upsets are found to alter the relative occurrence of false positives and false negatives in the classification of individual digits. This occurred despite the overall classification accuracy remaining unaffected.

CHAPTER II

BACKGROUND

Neuromorphic Computing

Neuromorphic computing is brain-inspired, non-von Neumann computing. The goal of neuromorphic computing is to emulate the way the brain approaches complex problems, such as pattern recognition. There are a variety of applications for neuromorphic computing including signal processing, high performance computing, text and audio processing, learning and optimization among others [1]. Interest in specialized hardware implementations to accelerate these computations has increased recently. Several companies including IBM, Intel, Qualcomm, and Nvidia have produced specialized hardware solutions implementing neuromorphic computing architectures in digital CMOS. These early generation products will ideally evolve into highly-integrated, three-dimensional, non-volatile technologies that provide low power processing, similar to biological brains.

In a similar manner to software neural networks, neuromorphic architectures are densely interconnected with large fan out to other neurons. Consequently, the architecture possesses increased redundancy compared to traditional computing architectures and can be fault tolerant if specifically designed that way, but comes at a computational cost [2-5]. The way neuromorphic computers "learn" is through the use of self-tuning weights. Since the memory and computing components of the chip are not physically separated as is the case in traditional architectures, the architecture is non-von Neumann. As a result, neuromorphic architectures have increased redundancy and a reduction in the memory bandwidth bottleneck compared to traditional

computing architectures [2]. While research is limited on how single events affect the relatively new architectures, studies have previously been conducted on realizations in COTs (Commercial off the Shelf), FPGAs (Field Programmable Gate Arrays), GPUs (Graphic Processing Units), and software frameworks [3, 5-7]. Several works in the literature analyze errors in neural networks through fault injection simulations [3-4, 6-9], radiation tests [3, 6-8], and pulsed-laser tests [10]. This work analyzes the effects of radiation on a neuromorphic computing architecture. Fault injection simulations were conducted in [8] based on the same architecture as this work.

This work uses image classification accuracy to gauge the effects of radiation in a neuromorphic computing architecture. With proton irradiation of a neuromorphic chip, we observed nearly unaffected overall classification accuracy, but offsetting trends in false positives and negatives in the recognition of individual image classes. This metric of image classification accuracy provides further insight into the radiation response.

Structure of Neural Networks

Neuromorphic computing architectures can be modeled using neural networks. There are several different types of neural networks, including artificial, convolutional, and deep neural networks to name a few. The neural network used for the experiments with TrueNorth is a 14-layer convolutional neural network.

Artificial Neuron

Artificial neurons are the basic building blocks used in the construction of the abovementioned neural networks [11]. These artificial neurons are a mathematical way to model biological neurons for use in neuromorphic architectures. Each input x_i to the neuron (a number) is weighted by its corresponding weight w_i, and the sum of all these weighted inputs is computed.

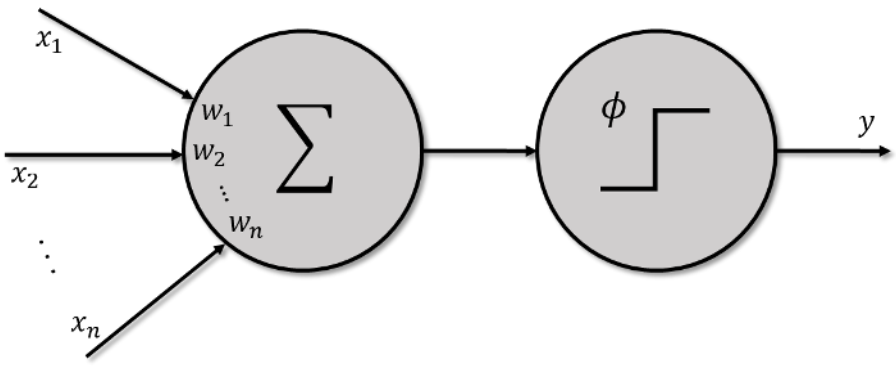

Fig. 1. Representation of an artificial neuron.

This weighted sum is then applied to an activation function φ which then results in the output y. The output y from the artificial neuron can then be used as an input to one or more neurons in the next layer. Equation 1 shows the formula governing the artificial neuron, while Fig. 1 is a representation of an artificial neuron. Artificial neurons can be implemented in software, digital circuits, or analog circuits.

$$y = \phi \sum_{i=1}^{n} w_i x_i \tag{1}$$

Activation Functions

The weighted sum of the inputs can take on a range of values. The purpose of the activation function is to map these weighted sums to one of the possible outputs. In the case of the activation function being a step function, values above a certain point produce a high output, while values below this point result in the output y being low. Thus, even though the weighted sum can take on a variety of values, there are only two possible outputs for a step activation function. Other activation functions exist and have various applications and complexity. Some additional activation functions include sigmoid and hyperbolic tangent functions, which have bounded

outputs; linear functions, whose outputs are unbounded; and functions that are not one-to-one, where different inputs can result in the same outputs.

Perceptron

One of the most basic neural networks composed of artificial neurons is the perceptron [11]. A perceptron is composed of a single layer of artificial neurons, and the most basic type of perceptron uses the step function as its activation function. As a result of using the step function, there are only two possible outputs: 1 or 0. The bias b is used in determining the output state of the perceptron. This bias is independent of the input value and is simply an offset from the origin. The perceptron output is 1 (high) when the neuron's weighted sum is greater than the bias, and the output is 0 (low) if the weighted sum is less than the bias. This single perceptron can be used for binary classification. For example, when given a picture of an animal, it could classify it as either "a dog" or "not a dog." Since the output is binary, it is unable to further refine its original "not a dog" classification to "a cat." More complex neural networks that use additional layers of neurons are necessary to achieve non-binary outputs.

Artificial Neural Networks

Artificial neural networks are created by connecting several perceptrons together. These networks have more processing power, flexibility, and applicability than lone perceptrons. Artificial neural networks are claimed to be intrinsically fault tolerant; however, most artificial neural networks cannot be considered fault tolerant without a proper design. Passive fault tolerance can be achieved through redundancy or by modifying learning [4-5]. Including faults during training promotes better generalization than other limited solutions, but it comes at both a higher computational cost and longer time for training [4-5]. Active fault tolerance such as resetting the

neural network to a faultless state after a fault occurs and propagates is the most widely used design of fault tolerant neural networks in hardware [4-5].

Feedforward Neural Network

A feedforward neural network is an artificial neural network where the perceptrons are linked together in multiple layers and does not consist of any cycles. This more complex neural network is composed of many neurons and can solve a wider variety of classification problems. Each layer can have any number of neurons in it, and the number of neurons in one layer has no bearing on the number of neurons allowed in subsequent layers. The first layer is called the "input layer" and receives the inputs to the neural network. The final layer is called the "output layer" since this layer produces the final output of the neural network. All layers (if any) between the input and output layers are called "hidden layers" because their inputs and outputs are not directly observable outside the neural network [12]. The number of hidden layers can vary depending on the application and the complexity needed in the neural network. If every neuron in one layer is connected to every neuron in the next layer, the layers are considered to be "fully connected." Fig. 2 is a representation of a fully connected feedforward neural network that contains only one hidden layer.

Convolutional Neural Network

A convolutional neural network (CNN) is a neural network composed of perceptrons where convolution is performed instead of matrix multiplication in at least one layer. A typical configuration will have one or more convolution layers at the beginning with subsequent layers being fully connected, matrix multiplication layers as in a standard multilayer neural network. The convolution aspect of a CNN is able to take advantage of the two-dimensional nature of some inputs, such as audio files and images. CNNs have several advantages over fully connected neural

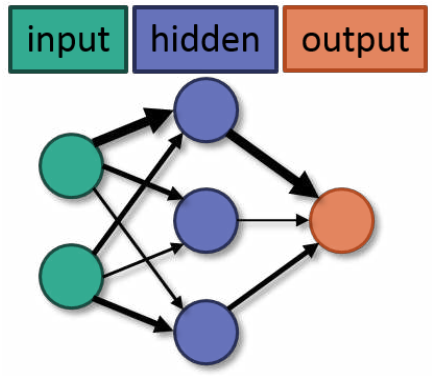

Fig. 2. Fully connected feedforward neural network with a single hidden layer and various weights between the neurons.

networks including easier training and fewer parameters per hidden nodes [13]. The neural network trained for use in this work utilizes a 14-layer CNN and classifies handwritten digits from the MNIST database. The information in the 784-pixel input image is given to the input layers, processed through the 12 hidden layers, and the final output is given in the form of 10 output neurons which contain information about the likelihood of the image being digits 0 through 9.

Radiation Effects

Radiation can be found in several environments, one of which is space. Protons, alpha particles, and heavy ions can produce SEEs, and these particles can be found in Earth's radiation belts, galactic cosmic rays, and solar heavy ions. Galactic cosmic rays are predominantly composed of proton and alpha particles, with less than 1% of the total particles being heavy ions. Solar heavy ions are produced by coronal mass ejections or solar flares [14].

A single event effect (SEE) may occur when a single ionizing particle strikes a semiconductor device. As this particle makes its way through the device, it leads to the creation of electron-hole pairs, depositing charge along its path [15]. Possible ionizing particles that can lead to an SEE include heavy ions, alpha particles, neutrons, and energetic protons.

A single event upset (SEU) occurs when an SEE changes the state of the device. SEUs are problematic since bit flips cause corruption of the stored information. These SEUs can propagate through a circuit or system resulting in errors in the output [16]. In occurrence with Moore's law, as transistor density increases, distance between transistors decreases [17]. As a result, an ionizing particle can strike more than one transistor resulting in multiple SEUs, particularly in small technology nodes [18].

Fault Tolerance of Neural Networks

In the context of computing systems, fault tolerance is defined by systems which still produce accurate results or outputs despite the failure of some components within the system. Fault tolerance can be subdivided into two categories: active and passive fault tolerance. Passive fault tolerance is achieved by utilizing the system structure to reduce faults, of which some examples are fault masking and redundancy. On the other hand, active fault tolerance is obtained by doing something within the system while it is running to find and manage errors as they occur. Examples include retraining and error detection and correction (EDC) [19].

Typically, fault tolerance in these computing systems is obtained by error correction (active fault tolerance) and/or redundancy (passive fault tolerance) within [20]. However, merely duplicating all the components in the computing system or adding EDC throughout the whole system is not the most efficient means of producing a design that is fault tolerant. While doing so to every component would result in a fault tolerant system, there is a high computational cost by the addition of all these elements. Therefore a balance is necessary between adding enough redundancy or EDC of important components and subsystems for high fault tolerance while still keeping a reasonable computational cost [20].

Digital Neural Circuits

Reference [11] examines the fault tolerance of artificial neural networks implemented on digital circuits using both fault injection simulation and SEU experiments. The fault injection simulations were conducted by flipping individual bits in the memory to mimic upsets caused by radiation due to single event effects. After modifying the memory, the artificial neural network was evaluated. These simulations showed that the neural networks tolerated a large percentage of the memory being upset. Between 86% and 99% of faults injected into the memory had no effect on the output. In addition, actual radiation experiments were performed. As in simulation, radiation had only a small effect on the output. Through the radiation experiments, it was also found that increasing the number of hidden layers in the artificial neural network increased the effect radiation had on the network. Both the simulation and experimental data showed that artificial neural networks gradually get worse with radiation instead of abrupt failures due to radiation.

Neural Networks Implemented on FPGAs and GPUs

Fault injection simulations and radiation tests have been conducted on FPGAs and GPUs among others. In simulations on an FPGA in [3], it was shown that faults in the neural network did not necessarily result in an error in the output. As such, faults that do not affect the output are called tolerable errors, hidden errors, or silent data corruption, while faults impacting the output are called critical errors. Additionally using fault injection in an FPGA, [7] showed that faults injected close to the input layers of the neural network had a more significant effect on the output than upsets in later layers. It was found in [6] that a single fault in a GPU tends to spread through multiple threads, and therefore it is important to be able to detect and correct errors in critical applications. Finally, [4] shows that deep neural network resistance to errors is dependent on a

number of factors including on the data itself in addition to the types of layers in the neural network design.

SRAM-Based Artificial Neural Networks

In [10], an SRAM-based artificial neural network tasked with image recognition was hit with a pulsed laser to generate SEUs. The beam was focused onto cells in the SRAM that contained weights and thresholds for the artificial neural network. In most cases, the image recognition rate did not change, indicating that the errors did not affect the output. In the cases that the recognition rate did change, it was improved in some cases and diminished in others. The experiments with the pulsed laser affected artificial neural network performance in a manner that error injection through software simulation could not [10].

CHAPTER III

TRUENORTH

Hardware

The particular hardware studied in this work was IBM's TrueNorth Neurosynaptic System fabricated in Samsung's 28nm low power (LP) process. It is programmable and executes trained "corelets," which are TrueNorth programs uploaded to the board via scripts from a computer. The chip implements a spontaneously-spiking neural network (SNN), meaning that the output spikes are not governed by a global clock. The neurons in the SNN integrate the input spikes and add to the membrane potential. This potential takes into account the weights of the neurons, and the weight decreases as a function of time, causing newer events to have more effect than older ones. When the membrane potential surpasses the pre-determined threshold, the output neuron fires a spike into the network. In addition, the SNN in the TrueNorth chip also includes leakage from the potential to emulate "forgetting" [21]. Therefore, older events have less of an effect on the neural potential than more recent events [22].

The chip was designed to be power efficient. The power efficiency is due mainly to local, distributed memory that limits the distance on the chip that spikes must travel. TrueNorth also uses an event-driven architecture, the SNN, therefore components only have to be "on" and consuming power when prompted by an event. The lack of a global clock network also contributes to the power efficiency of the chip [21].

TrueNorth is scalable and fault tolerant [21]. Up to sixteen TrueNorth chips can be physically connected, and the software is already designed to allow communication among the

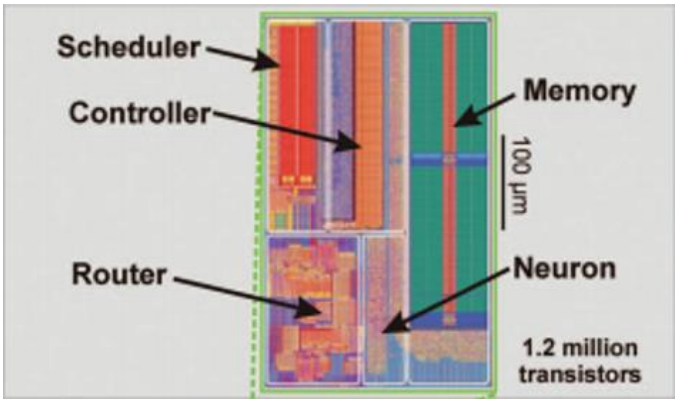

Fig. 3. Schematic of one of the 4096 cores in a TrueNorth chip. Courtesy of IBM [21].

various chips. In addition, the high connectivity between neurons, cores, and chips creates redundancy. A result of the interconnectedness is increased fault tolerance via avoidance of defective cores and the implementation of memory redundancies [21].

The physical arrangement of one TrueNorth chip is a 64 by 64 array of cores (4096 total cores) covering an area of 4.3 cm^2 including a total of 428 Mb of SRAM storing neuron data [21, 23]. A schematic of one core is shown in Fig. 3. The total amount of SRAM per core is approximately 10^7 kb [23]. The values for the neuron magnitudes and configuration weights are stored in the SRAM as digital outputs.

The packaging of the TrueNorth chip includes 1.2 mm molding compound above the chip, as shown in Fig. 4. Below the molding compound, approximately 10 µm of metallization, dielectric, and via layers exist above the active region. Fig. 5 shows a sketch of the back end of line (not to scale). These interconnect layers are composed of various metals including aluminum and tungsten, possibly along with copper.

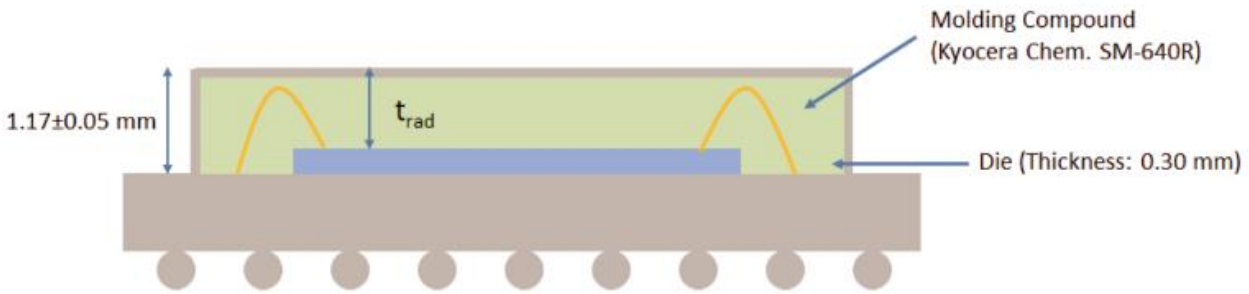

Fig. 4. Illustration of the TrueNorth package (not to scale). Courtesy of UCLA.

Operation

The TrueNorth board operates on trained corelets. The particular corelet loaded in for these experiments was pre-trained where the training was conducted based on an IBM model file. The corelet classifies handwritten digits 0 through 9 from the modified NIST (MNIST) database. Of the 70,000 digits in the database, there are 60,000 digit images for the training data set and 10,000 for classification. Of the 10,000 digits used for classification, there are approximately 1000 samples of each digit [24].

The classification of a digit is signaled by the output neuron magnitudes. For an input image, the corelet evaluates 10 output neuron magnitude values ranging from 0 to 1 inclusive. An output neuron magnitude can be thought of as the likelihood of a particular digit being the correct classification of the image. Note, however, that the sum of all output neurons is not required to add up to 1 like independent probabilities. The output classification reported is the digit with the highest output neuron magnitude.

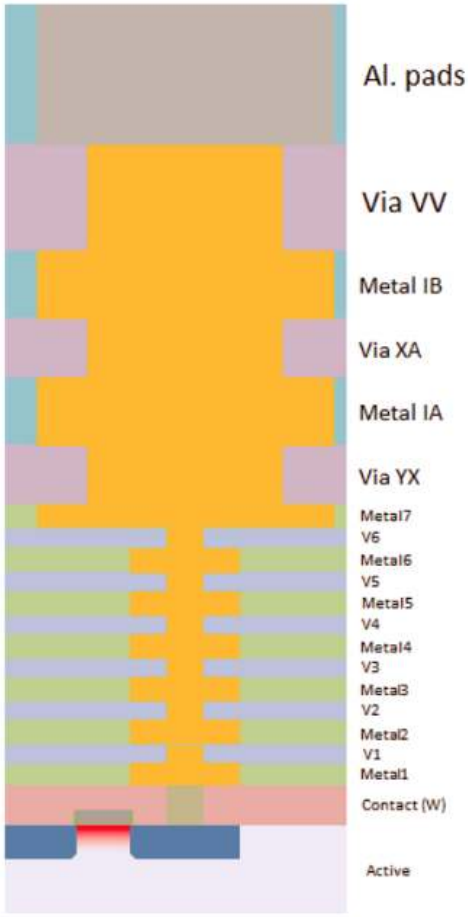

Fig. 5. Schematic cross section of the IBM TrueNorth chip. The approximate height of the back end stack is about 10μm. Courtesy of UCLA.

CHAPTER IV

EXPERIMENTAL SETUP

Vanderbilt's Pelletron is an electrostatic particle accelerator primarily used to conduct radiation tests on electronic devices. The primary particles that can be accelerated are protons, alpha particles, oxygen ions, and chlorine ions, but it is possible to use other ions as well. The selected ion is accelerated using a positive electrostatic potential. Several nanometer-scale gold foils are used to scatter the beam, and magnets are used to steer the beam. The test chamber is cylindrical with an approximately two-foot diameter and height of 20 inches. The mount on which the electronics are placed is adjustable to provide different beam incident angles for the tested device. Additionally, there are electrical feed-throughs to allow for testing under bias and for real-time data acquisition [25].

Chip Preparation

The molding compound above the TrueNorth chip (Fig. 4) had to be removed to reduce the range necessary for the ions to reach the sensitive area. A picture of the exposed chip is shown in Fig. 6.

Minimum Ion Energies

As previously mentioned, above the active region of the TrueNorth chip is approximately 10 μm of back end of line materials. As a result, the incident ion must have sufficient range to reach the sensitive volume to produce a SEE within the chip. Table 1 shows the minimum energy

Fig. 6. Picture of the TrueNorth board after the chip (middle-left) was delidded.

Table 1. Minimum energy required for various ions to penetrate 10μm of various metals.

minimum energy (MeV)	metal					
incident ion (maximum ion energy)		**Al**	**Cu**	**Si**	**SiO_2**	**W**
	proton (4 MeV)	0.8	1.4	0.8	0.65	1.6
	alpha (6 MeV)	3	5	2.75	2.25	6
	oxygen ion (14.3 MeV)	>14.3	>14.3	>14.3	12	>14.3
	chlorine ion (16.4 MeV)	>16.4	>16.4	>16.4	>16.4	>16.4

required for the incident ion to penetrate a bit less than 10 μm of various metals as calculated by SRIM [26]. The maximum energy that can be achieved by the Pelletron varies based on the ion chosen. In the table above, if the energy required for a particular ion to penetrate around 10 μm of metal is greater than the maximum energy for that ion, the data point is represented as greater than the maximum Pelletron energy for that ion and grayed out. Using the SRIM energy data from Table 1, the possible ions to use for the SEE testing on the TrueNorth chip was narrowed down to either protons or alpha particles. If the metallization layers are composed of certain materials, there might not be enough energy in the oxygen ions or chlorine ions for SEE testing. It was also noted from the table that the minimum energy required to penetrate the metallization layers varies for

Table 2. Range of various ions at the Pelletron's maximum energy of those ions in various target materials.

range of ion (μm)		metal				
		Al	**Cu**	**Si**	**SiO_2**	**W**
incident ion (maximum ion energy)	**proton (4 MeV)**	130	54	150	180	40
	alpha (6 MeV)	28	13	32	38	10
	oxygen ion (14.3 MeV)	8.8	4.3	9.9	12	3.4
	chlorine ion (16.4 MeV)	5.7	3.0	6.3	8.1	2.4

different materials. Silicon dioxide has the lowest minimum energy, followed by silicon, then aluminum, copper, and finally tungsten.

Maximum Ion Ranges

In addition to looking at the minimum energy required for various ions to travel a certain distance through a material, SRIM is also able to calculate the range of ions in different materials. Table 2 is a range table of ions in various metals. To penetrate the metallization layers, the range must be at least 10 μm, therefore data points that do not achieve this specification are grayed out. Both protons and alpha particles have sufficient range for the various target materials considered. Oxygen ions are not guaranteed to have sufficient range to penetrate the metallization layers, particularly in areas that have high concentrations of copper or tungsten. Chlorine ions also do not have enough range for the energy at which they can be tested. Protons were selected since they were found to have sufficient energy and therefore range for SEE testing on TrueNorth.

Testing Setup

The radiation tests were conducted at Vanderbilt University using the Pelletron. The proton energy selected was 4 MeV, and at that energy, the possible proton fluxes range from 10^5 to 10^{10} protons/cm^2-s [25]. The TrueNorth board (Fig. 6) allowed for trained corelets to be uploaded via scripts from a computer. During irradiation, the board was mounted in the Pelletron's vacuum chamber, and an Ethernet feed-through from the chamber was connected to the board.

CHAPTER V

EXPERIMENTAL RESULTS

Pre-Irradiation Accuracy

The accuracy of pre-irradiation runs on the 10,000 digit MNIST test set using a pre-trained, fixed network (corelet) is 98.82%, which corresponds to 118 errors of the 10,000 digits classified. This is the baseline classification accuracy. Since TrueNorth operation is deterministic, all runs without radiation yield this exact accuracy. (Note these errors are not due to radiation, but rather the inherent nature of a finite, imperfect training set and occasionally ambiguous, sloppily handwritten digits.) A sample of some digits classified by the trained corelet are shown in Fig. 7. The green boxes in the figure represent examples of correct classifications. The red box is an example of an incorrect classification, which can also be called a critical error since this is a change in the output. The red box shows an example of a "9"; however, the trained corelet incorrectly classifies this digit as a "4" pre-irradiation.

Post-Irradiation Accuracy

In previous research on similar architectures emulated using FPGAs and software, parts were irradiated to a certain fluence, and the total number of errors were determined [5]. A similar experiment was run using the TrueNorth architecture. Multiple exposures were performed with fluences up to 5.6×10^7 cm^{-2} followed by determination of the overall classification accuracy. Between runs, the chip was reprogrammed and verified to consistently exhibit pre-irradiation

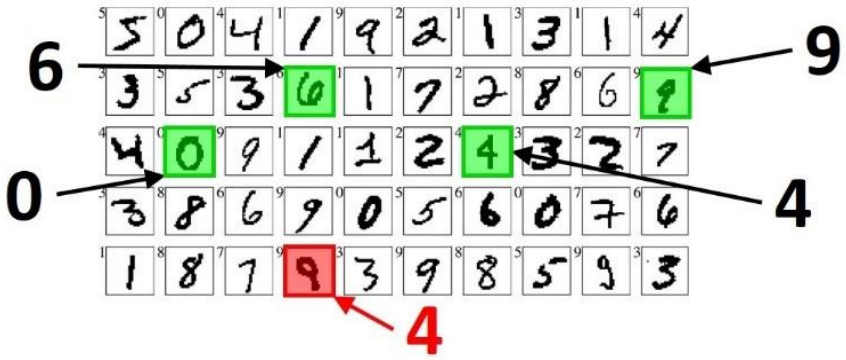

Fig. 7. Example of digits classified by the MNIST corelet running on the TrueNorth chip. Green indicates correct classification while red indicates an incorrect classification. Adapted from [29].

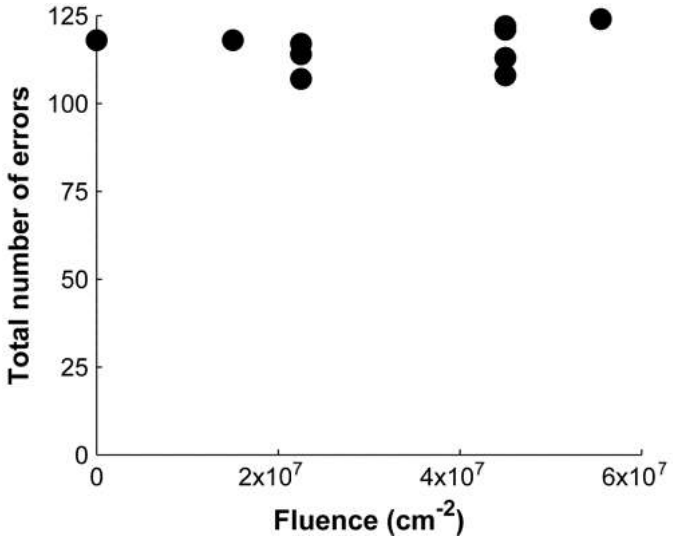

Fig. 8. Number of classification errors (out of 10,000) as a function of fluence after irradiation.

performance, including across classes. The results of the collection of runs from this experiment are shown in Fig. 8, where the number of classification errors is plotted as a function of fluence. The accuracy remains about the same after irradiation with only small variations with respect to the pre-irradiation, baseline number of errors. For the nine runs in Fig. 8, the mean number of errors is 116 errors, and the percent standard deviation is 5%.

Mechanisms

The errors resulting from irradiation are not permanent; they are soft errors. Reloading the model file returns TrueNorth to the original, pre-irradiation state. In addition, classification of a set of post-irradiation data results in the exact same classification every time. TrueNorth's classification of the data is deterministic. As long as new errors are not introduced via irradiation, the resulting classification of all 10,000 digits remains the same each time. This is consistent with fault injection simulations on this architecture [8] and the expectation that the observed effects are the result of single event upsets in the SRAM.

The classification changes resulting from radiation occur due to single events, not total ionizing dose. For one, reloading the model file returns TrueNorth to its original, pre-irradiation state and classification. Additionally, the dose accumulated over all the exposures to the TrueNorth chip was found to be less than 1 krad(SiO_2). This provides further evidence that single event effects, not total ionizing dose, are causing the changes in classification.

Critical Errors: Classifications Changed by Radiation

Even though radiation did not significantly alter the total number of classification errors, some of the output classifications changed due to proton-induced soft errors, with the number of changes increasing with fluence. A pictorial example of how this can happen is shown in Fig. 9. The top row shows the handwritten digits 6, 7, 8, and 9 that are input into the trained corelet for classification. The bottom row shows the classification resulting from the corelet. The left side of the arrow shows the pre irradiation results, and the right side shows the results after irradiation to a proton fluence of 5.6×10^7 cm^{-2}. Green boxes signify when the classification was correct, while red corresponds to an incorrect classification. In this simplistic example, the classification

Fig. 9. Example of how changes can occur due to radiation without affecting the overall classification accuracy. The left side is the pre-irradiation classification of the digits 6, 7, 8, and 9, and the right side is the same digits after irradiation to a fluence of 5.6×10^7 cm^{-2}.

accuracy both before and after irradiation is 75%. However, when comparing the outputs from the two classifications, there were actually two changes in classification. The "8" that was originally classified as a "5" was corrected, while the "7" (originally classified correctly) was incorrectly classified as a "3" after irradiation. So while the overall classification accuracy remained the same before and after irradiation (75%), there were actually two changes in classification resulting from radiation. Using this technique of comparing the results of the pre-irradiation to the post irradiation classification, Fig. 10 is generated using the same experimental runs used in Fig. 8. This figure shows the number of changes in classification compared to pre-irradiation as a function of fluence, where zero changes corresponds to the pre-irradiation classifications. As fluence increases (corresponding to an increased number of soft errors in the SRAM), the number of classification changes increases. However, as described before, there are approximately the same number of classification corrections (like "8" in Fig. 9) as incorrect classifications due to soft errors (like "7" in Fig. 9). In addition, there are some instances where an incorrect classification is not corrected, but the resulting output changes. An example of this is shown in Fig. 11. Low confidence classifications were most susceptible to classification changes after irradiation.

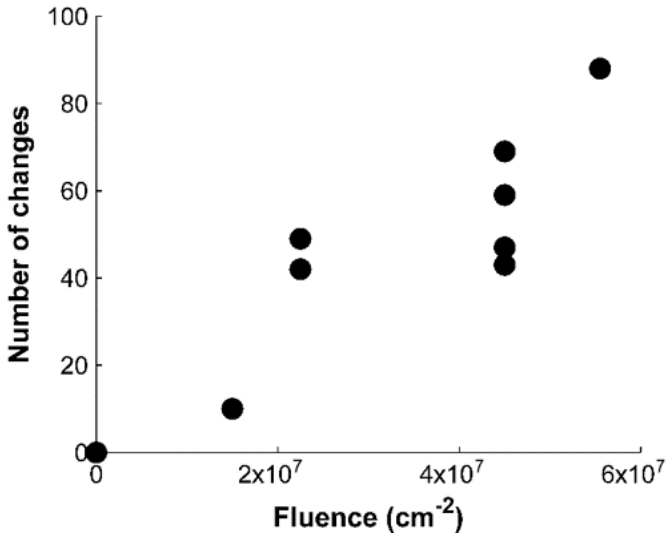

Fig. 10. Number of changes in classification due to radiation with respect to original, pre-irradiation classification as a function of fluence.

Fig. 11. Example of a digit classification changing due to irradiation, but still being incorrectly classified. Pre-irradiation (left), the 5 is incorrectly classified as a 9, and then after irradiation to a fluence of 5.6×10^7 cm^{-2}, the same digit is incorrectly classified as a 3 (right).

Tolerable Errors: Output Neuron Magnitudes Changed by Radiation

As previously mentioned, the output neuron magnitudes of the trained corelet determine the resulting classification of an image, and these values are stored in the SRAM. There are 10,000 MNIST images being classified per run and each image has 10 output neuron magnitudes, so there are a total of 100,000 output neuron magnitudes per run. Because the output neuron magnitudes are not the final output, changes to these output neuron magnitudes are considered to be tolerable errors.

After irradiation to a fluence of 2.6×10^7 cm^{-2}, 29% of the output neuron magnitudes had changed due to SEUs in the SRAM. However, despite the large percentage of changes in output neuron magnitudes, there were still only 118 classification errors (98.82% accuracy), which is the same total number of errors pre-irradiation. Analysis showed 39 classification changes between the pre-irradiation and post-irradiation data, but for every classification error due to radiation, there was a correction. Of the 39 changes after radiation, 17 of the changes resulted in classification errors, 17 were classification corrections, and the remaining 5 were classifications that were incorrect both before and after irradiation.

In order to cause a change in classification, the value of one output neuron magnitude must overcome the output neuron magnitude of the pre-irradiation classification. Fig. 12 shows the output neuron magnitude changes due to radiation that resulted in the 17 classification errors while Fig. 13 shows the 17 classification corrections. In both figures, the x-axis is the output neuron magnitude. The changes in output neuron magnitudes are represented by arrows. The start of the arrow corresponds to the value of the pre-irradiation output neuron magnitude, and the termination of the arrow corresponds to the post-irradiation value of the output neuron magnitude. The pairing of a red and black arrow represents the output neuron magnitudes for a single image. The red, dashed arrow represents the output neuron magnitudes for the incorrect classification. Conversely, the black, solid arrow corresponds to the output neuron magnitudes of the correct classification. Black digits are the values of the correct classification while red digits are the values of the incorrect classification.

Since the value of the output neuron magnitude indicates the correlation between the output neuron magnitude's digit and the input image, output neuron magnitudes on the right side of the graph indicate high confidence in the resulting classification, while those on the left suggest low

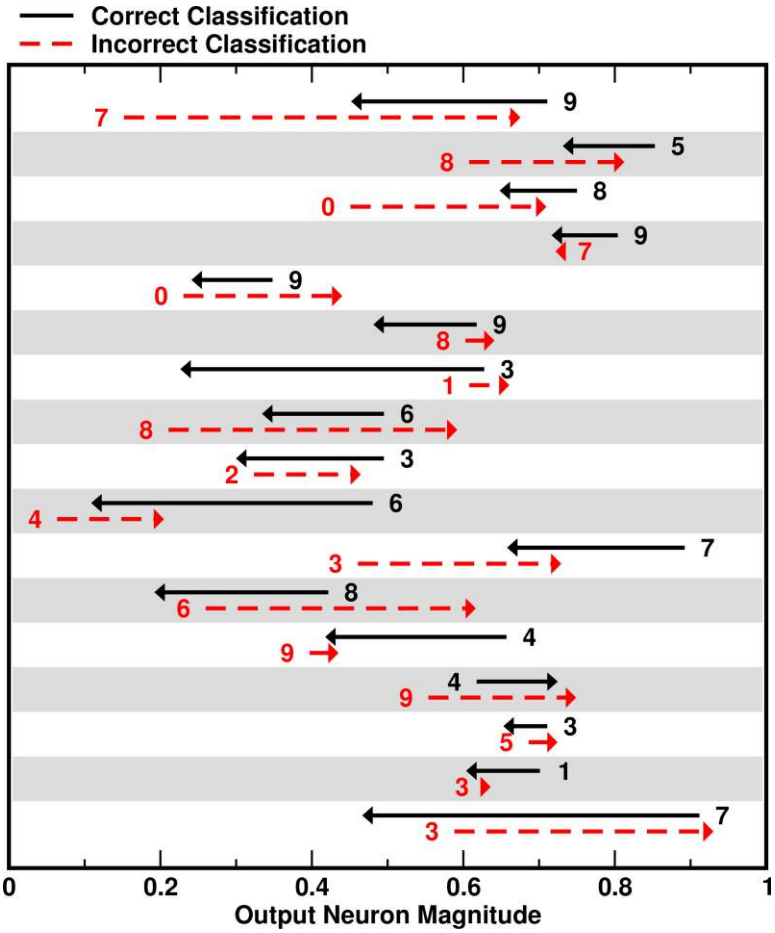

Fig. 12. Output neuron magnitudes for 17 classification *errors* resulting from radiation. Black arrows and digits correspond to the correct classification while red arrows and digits correspond to the incorrect classification. The beginning of the arrow is the output neuron magnitude pre-irradiation and the termination of the arrow is the post-irradiation output neuron magnitude.

confidence. If high confidence is considered to be output neuron magnitudes with values greater than 0.95, then 93% of pre irradiation classifications on the 10,000 images set were found to be high confidence. In addition, of these high confidence classifications, 73% of them had an output neuron magnitude value of exactly 1. Figs. 12 and 13 show that the classification errors due to irradiation resulted from output neuron magnitudes which are not classified as high confidence. None of the classification changes resulted from high confidence output neuron magnitudes, and

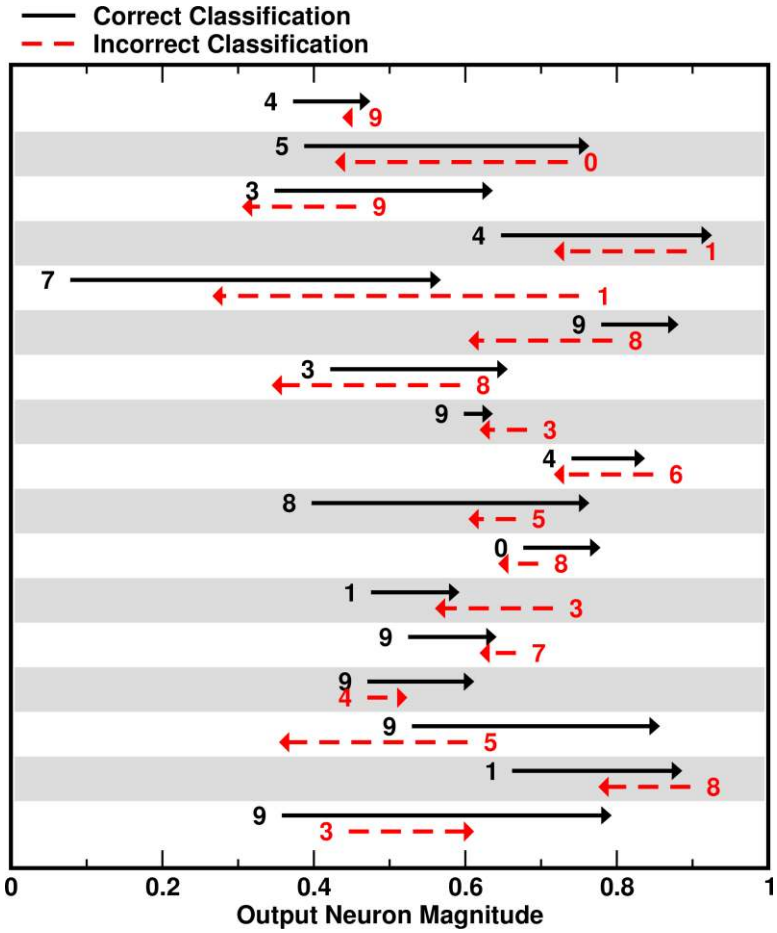

Fig. 13. Output neuron magnitudes for 17 classification *corrections* resulting from radiation. Black arrows and digits correspond to the correct classification while red arrows and digits correspond to the incorrect classification. The beginning of the arrow is the output neuron magnitude pre-irradiation and the termination of the arrow is the post-irradiation output neuron magnitude.

only a few of the classification changes resulted from pre-irradiation output neuron magnitudes above 0.8.

CHAPTER VI

SPECIFIC CLASSES

In addition to analyzing the overall errors and classification changes, the effect of soft errors in the SRAM on specific digits is analyzed. There are two ways of looking at every classification error that occurs. Fig. 14 shows a sample string of inputs to the corelet and the corresponding output classification. As seen in the figure, there is a single error, the "8" being incorrectly classified as a "5". The "8" is an example of a *false negative*, where a false negative is not classifying the input correctly. On the other hand, the "5" is a *false positive*, which is an indication of an incorrect result. In other words, *false negative* corresponds to the input that is incorrectly classified, while *false positive* corresponds to the erroneous output. While a false negative is the incorrectly identified digit (the "8" in Fig. 14), a false positive is the classification of a digit that is not actually there (the "5" in Fig. 14).

Here are false positives and false negatives in a more general context. A false positive is falsely alerting to something, or "crying wolf." On the other hand, a false negative is not alerting to something that it should. This means that something slips by and is "invisible" to the classification system. Both of these situations could be undesirable. For example, say that classifying something as X causes alarm. On one hand, every false positive for X produces unnecessary alarm since X did not really occur. On the other hand, every false negative for X means that there is no alarm when there rightfully should be, because X did occur even though the classification system did not classify it as such.

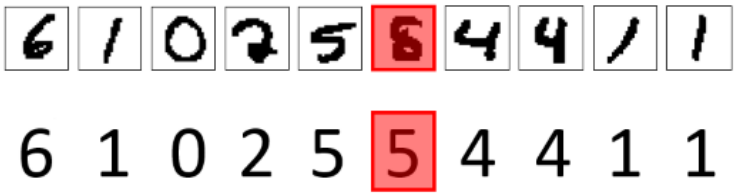

Fig. 14. Demonstration of how one error (an "8" incorrectly classified as a "5") can be separated into two components: a false negative (the "8") and a false positive (the "5".) The top row is the input, and bottom row is the resulting output classification.

Digit	Number of False Negatives	Number of False Positives
0	2	10
1	10	6
2	5	4
3	9	20
4	29	5
5	16	11
6	13	5
7	7	12
8	9	14
9	18	31

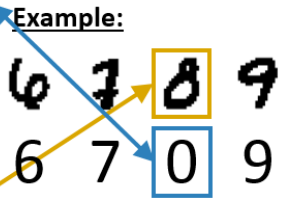

Fig. 15. The figure shows an example of an error and its corresponding false positive and false negative in the table. The table shows the number of false negatives and positives for each MNIST digit from 0 to 9.

Pre-Irradiation

Analyzing the pre-irradiation data to look at the false positives and negatives of specific digits results in Fig. 15. In addition, the figure shows an example of an error and how the false positive and negative relates to the table. Looking at the yellow box, that there are 9 instances that an "8" is incorrectly classified. For reference, there are approximately 1000 examples of the digit "8". On the other hand, there are 10 cases where the number input to the corelet is classified as a "0" when some other digit is actually input into the system. Since false negatives and false

positives each represent a different way of looking at one error, the sum of the false negative column is equal to the sum of the false positive column. These two sums are both 118, which is to be expected since that is the total number of errors in the system pre-irradiation.

It can also be seen from the table in Fig. 15 that there is a bias toward certain numbers. Of the ten digits, "0" has the lowest false negative classification rate. Of the approximately 1000 examples of "0" input into the system, only 2 were classified as something else. On the other hand, "4" has the highest instance of classification errors as shown by 29 false negatives. As far as false positives (the incorrect output classifications) go, "4" has only 5 instances, meaning that there are only 5 times that some input is incorrectly classified as a "4". In this case, there is a bias against classifying "4"s, since "4" has a high number of missed classifications (false negatives) and a low number of incorrect classifications (false positives). There is a bias towards classifying numbers as "0", indicated by the low number of false negatives and moderate number of false positives. These observations may depend on the training data set.

Post-Irradiation

The effect of radiation on the number of classification errors for specific digits is analyzed next. Figs. 16 and 17 show how the number of false positives changes with fluence for "0" and "6" respectively. As the fluence increases for the digit "0", the number of false positives decreases. On the other hand, the number of false positives for "6" increases with fluence.

Taking all the false positive and false negative data from figures like Figs. 16 and 17 and combining them into one histogram results in Fig. 18. Fig. 18 has the false positives at the top half of the y-axis and the false negatives going down the lower half of the axis. The pre-irradiation data

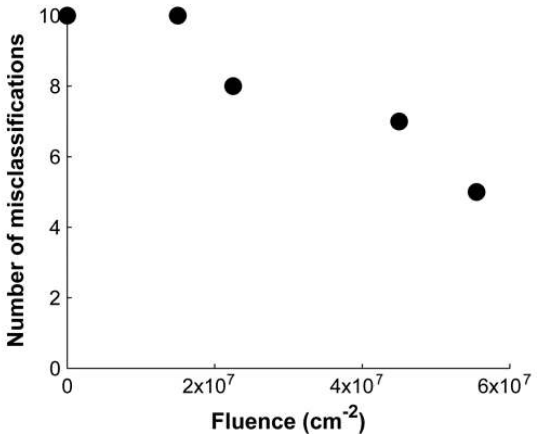

Fig. 16. As fluence increases, the number of false positives for the digit "0" decreases.

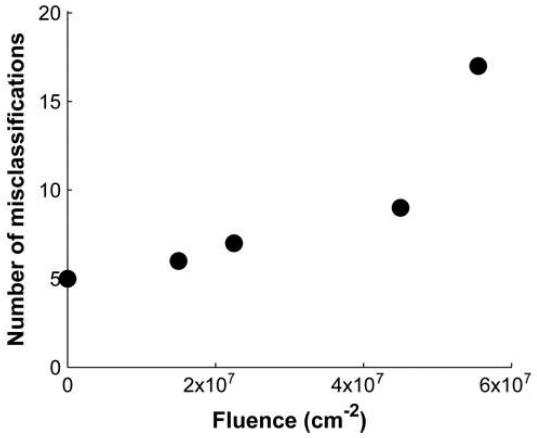

Fig. 17. As fluence increases, the number of false positives for the digit "6" increases.

sets are the yellow bars, while the blue bars are the post-irradiation data. The positive x direction corresponds to increasing fluence within each digit's data set.

This figure allows comparison between digits and also analysis of trends within a specific digit. The digit with the highest occurrence of false positives is "9", meaning that there are more incorrect outputs that are "9"s than any other digit. The highest number of false negatives belongs to the digit "4", meaning that the "4"s input into the system are the most often incorrectly classified digit. The number of false positives for "0" and "5" decreases with fluence, while it increases for

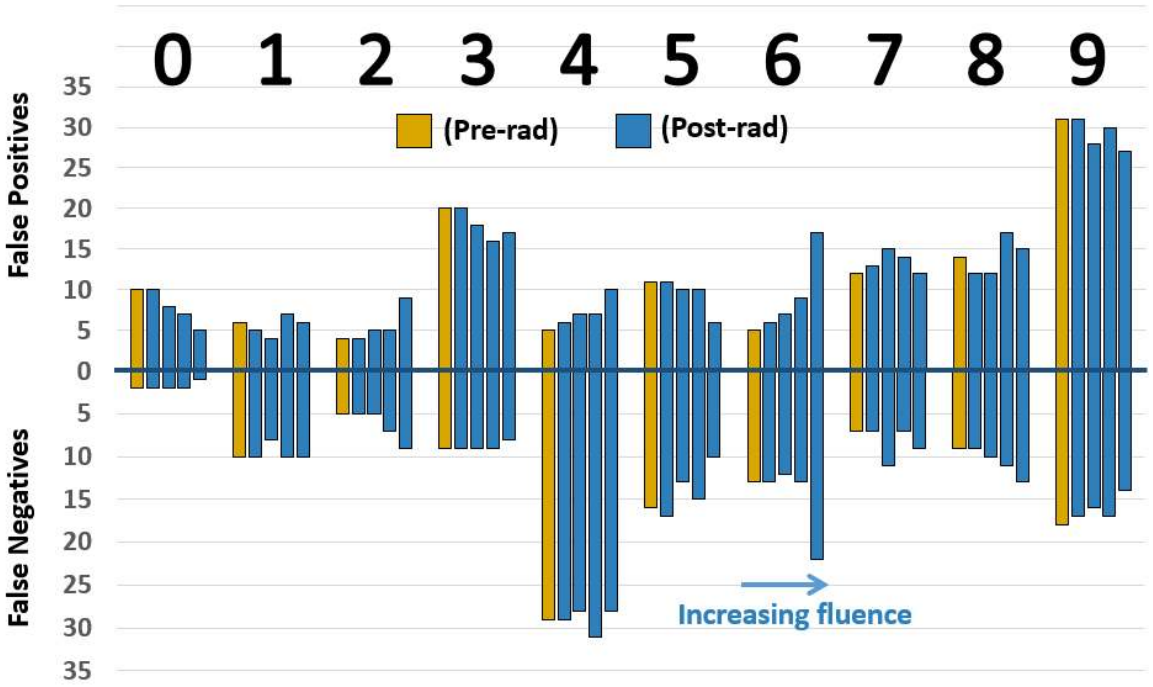

Fig. 18. Plot of false positives and false negatives for all digits versus increasing fluence.

"2", "4", and "6". On the other hand, the number of false negatives increases with fluence for the digits "2" and "8".

CHAPTER VII

SIMULATIONS

Neural Network Set Up

In this work, simulations were conducted on neural networks classifying the digits in the MNIST database. Keras [28], written in Python, is a high-level neural networks API which was used to create the simulations. Keras was used in conjunction with Tensorflow [29], an open-source framework for developing machine learning applications. Using Keras and Tensorflow, a script was written in Python that trained a neural network using a perceptron model and the MNIST training database and then evaluated the network on the MNIST classification set.

The neural network created was a multi-layer network consisting of an input layer, hidden layers, and an output layer. The first layer took as inputs and processed the 784 pixels in the 28x28 pixel MNIST image. This input layer contained no parameters (analogous to weights) and had 784 outputs, which were used as inputs to the hidden layers. The hidden layers used 12560 parameters to reduce the 784 inputs to just 16 outputs. These 16 outputs were then used as inputs to the output layer of the network. This final layer output 10 values which correspond to the confidence level for each digit 0 through 9. These 10 outputs are analogous to the output neuron magnitudes resulting from TrueNorth. The highest value of the 10 outputs corresponds to a digit, and that digit is the final classification. Just like in the TrueNorth architecture, the accuracy of the trained perceptron-based neural network was calculated, and the accuracy of the simulated neural network was 93.2% before fault injection.

Fault Injection

In order to simulate the effects of radiation on the neural network, the parameter values (weights) within the layers were changed after training. There were a total of 12730 parameters, the majority of which were contained in the hidden layers. To simulate a random upset in a neural network, a weight would be selected at random and then its value changed to a random new one within the range of possible weights. After upsetting a percentage of the total parameter values, the neural network was run on the classification set of MNIST digits.

After evaluation, the total accuracy of the classification was calculated. In addition, the number of changes in output compared to the "pre-irradiation" classifications was found. Then additional parameters were changed, the network evaluated, and the accuracy and number of changes calculated. This was repeated until 20% of the parameters had been changed. The plots in Figs. 19 and 20 were produced by seeing how the percent of total parameters upset affected accuracy and number of classification changes respectively.

Overall, Fig. 19 shows the gradual degradation of the classification accuracy as the number of injected errors increases. Sudden drops in classification accuracy can be attributed to injected errors that propagated to effect several outputs. Sections where the classification accuracy stayed approximately constant indicate that injected errors caused little change in the outputs. Finally, instances where the classification accuracy increased as the number of injected errors increased show that the simulated upsets caused classification corrections. Injecting errors into approximately 1% of the weights had very little effect on the classification accuracy. There is a

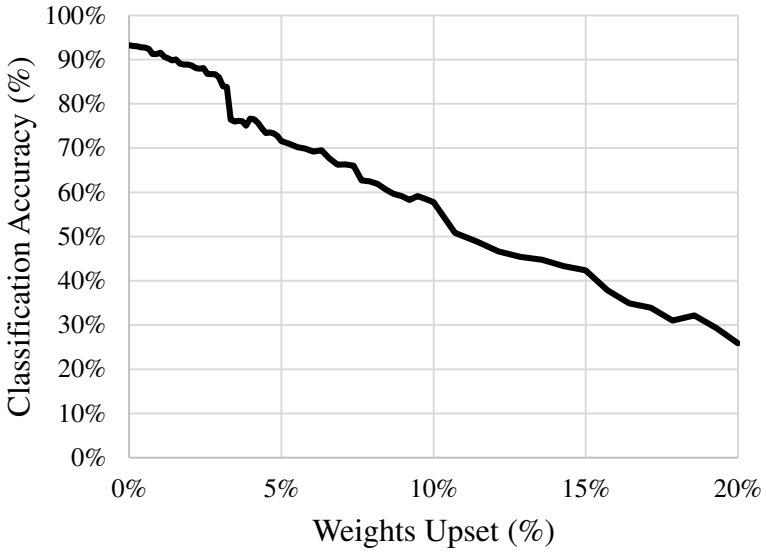

Fig. 19. Classification accuracy degradation as percentage of upset weight values.

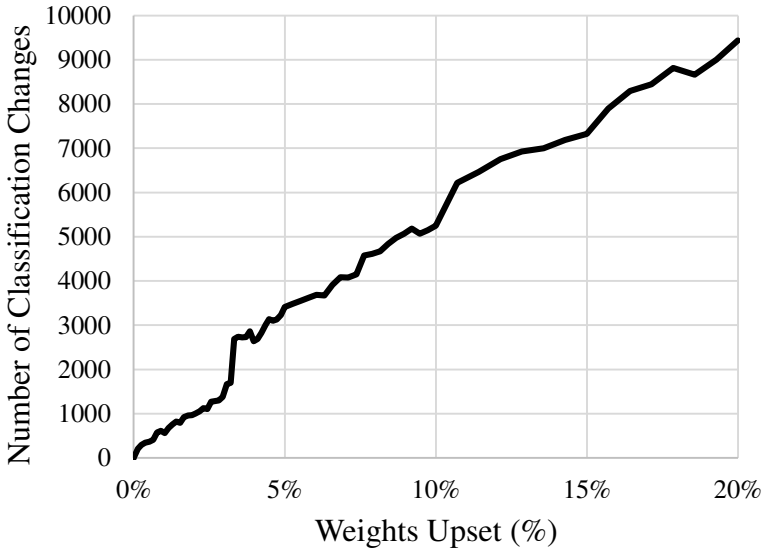

Fig. 20. Relative number of changes in classification as percentage of weight values upset.

slow degradation in accuracy until about 3% of the weights are upset, after which a jump in accuracy degradation is seen. It is also seen here that the accuracy of the neural network degrades to 50% accuracy after approximately 11% of the total weights are upset. In addition, when 20% of the total weights are upset, the accuracy of the system has dropped to only 25% accurate.

Fig. 20 shows the relative number of changes in classification resulting from the injected errors with respect to the original classification of the network. This is similar to comparing the post-irradiation classifications to the pre-irradiation classifications. The number of relative changes is zero (origin) when no weights have been modified since the neural networks are still exactly the same. It takes changing about 10% of the parameter values for half of the 10,000 MNIST images classifications to change. Additionally, after injecting errors into about 20% of the weights, almost every single classification had been modified.

CHAPTER VIII

CONCLUSIONS

The results of this experiment can be applied more generally to neuromorphic architectures. In the data presented here, the overall classification accuracy remained constant after introducing errors in the SRAM by proton irradiation; however, changes occurred within the specific digits. In neuromorphic computing architectures, it may be necessary to look not only at the overall accuracy but also the accuracy of specific classes. Even if the accuracy is not affected by single events, if a classification system gets better at classifying benign objects, but worse at recognizing critical objects, then arguably the classification system has worsened due to single events. Critical applications of neuromorphic systems may require a more in-depth analysis of the occurrence of individual errors.

Both the overall classification accuracy and individual digit classification accuracy were analyzed in this work. The classification accuracy before irradiation was 98.82%. Introducing soft errors by proton irradiation up to fluences of 5.6×10^7 cm^{-2} did not significantly affect this accuracy. Although the accuracy stayed approximately constant, tolerable errors did cause an increasing number of changes between pre-irradiation and post-irradiation classifications. In addition, when analyzing individual digits, these tolerable errors caused changes to the classification accuracy of particular digits. Some digits had a decrease in errors as fluence increased, while others had an increase in errors with fluence. As a result, analyzing the effect of tolerable errors on classification accuracy and individual digit classification accuracy may be a useful metric in evaluating neuromorphic architectures.

REFERENCES

[1] J. Sawada *et al.,* "TrueNorth Ecosystem for Brain-Inspired Computing: Scalable Systems, Software, and Applications," *SC16: International Conference for High Performance Computing, Networking, Storage and Analysis*, Salt Lake City, UT, 2016, pp. 130-141.

[2] I. A. Basheer and M. Hajmeer, "Artificial neural networks: fundamentals, computing, design, and application," in *Journal of Microbiological Methods*, pp. 3-31, Dec. 2000.

[3] F. Libano *et al.,* "Selective Hardening for Neural Networks in FPGAs," in *IEEE Transactions on Nuclear Science*, vol. 66, no. 1, pp. 216-222, Jan. 2019.

[4] G. Li *et al.,* "Understanding error propagation in deep learning neural network (DNN) accelerators and applications," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, Denver, CO, pp. 8:1-8:12, Nov. 2017.

[5] C. Torres-Huitzil and B. Girau, "Fault tolerance in neural networks: Neural design and hardware implementation," *2017 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, Cancun, 2017, pp. 58-63.

[6] F. F. d. Santos *et al.,* "Analyzing and Increasing the Reliability of Convolutional Neural Networks on GPUs," in *IEEE Transactions on Reliability*, vol. 68, no. 2, pp. 663-677, June 2019.

[7] F. Libano, P. Rech, L. Tambara, J. Tonfat and F. Kastensmidt, "On the Reliability of Linear Regression and Pattern Recognition Feedforward Artificial Neural Networks in FPGAs," in *IEEE Transactions on Nuclear Science*, vol. 65, no. 1, pp. 288-295, Jan. 2018.

[8] S. Moran, J. Cox, S. Iyer, R. Brewer and B. Sierawski, "Radiation Effects on Brain-Inspired Computing: A Study Utilizing the IBM TrueNorth Neurosynaptic System," in *GOMACTech*, Albuquerque, NM, Mar. 2019.

[9] A. Azizimazreah, Y. Gu, X. Gu, and L. Chen. "Tolerating Soft Errors in Deep Learning Accelerators with Reliable On-Chip Memory Designs," *IEEE International Conference on Networking, Architecture and Storage (NAS)*, Chongquin, China, pp. 52-61, Oct. 2018.

[10] S. Buchner *et al.,* "Pulsed laser validation of recovery mechanisms of critical SEEs in an artificial neural network system," in *IEEE Transactions on Nuclear Science*, vol. 45, no. 3, pp. 1501-1507, June 1998.

[11] R. Velazco, A. Assoum, N. E. Radi, R. Ecoffet and X. Botey, "SEU fault tolerance in artificial neural networks," in *IEEE Transactions on Nuclear Science*, vol. 42, no. 6, pp. 1856-1862, Dec. 1995.

[12] R. Reed and R. J. Marks, *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks,* Cambridge, MA: MIT Press, 1999.

[13] A. Ng *et. al.,* "Convolutional Neural Network," *UFLDL Tutorial*, Accessed on: Oct. 24, 2019. [Online]. Available: http://deeplearning.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/

[14] J. L. Barth, C. S. Dyer and E. G. Stassinopoulos, "Space, atmospheric, and terrestrial radiation environments," in *IEEE Transactions on Nuclear Science*, vol. 50, no. 3, pp. 466-482, June 2003.

[15] P. E. Dodd and L. W. Massengill, "Basic mechanisms and modeling of single-event upset in digital microelectronics," in *IEEE Transactions on Nuclear Science*, vol. 50, no. 3, pp. 583-602, June 2003.

[16] C. S. Guenzer, E. A. Wolicki and R. G. Allas, "Single Event Upset of Dynamic Rams by Neutrons and Protons," in *IEEE Transactions on Nuclear Science*, vol. 26, no. 6, pp. 5048-5052, Dec. 1979.

[17] G. E. Moore, "Cramming More Components onto Integrated Circuits," in *Electronics*, pp. 114–117, April 19, 1965.

[18] L. W. Massengill, B. L. Bhuva, W. T. Holman, M. L. Alles and T. D. Loveless, "Technology scaling and soft error reliability," *2012 IEEE International Reliability Physics Symposium (IRPS)*, Anaheim, CA, 2012, pp. 3C.1.1-3C.1.7.

[19] C. Torres-Huitzil and B. Girau, "Fault and Error Tolerance in Neural Networks: A Review," in *IEEE Access*, vol. 5, pp. 17322-17341, 2017.

[20] K. Mehrotra, C. K. Mohan, and S. Ranka, "Fault Tolerance of Neural Networks," by *Rome Laboratory*, Griffiss Air Force Base, NY, 1994.

[21] F. Akopyan *et al.,* "TrueNorth: Design and Tool Flow of a 65 mW 1 Million Neuron Programmable Neurosynaptic Chip," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and System*s, vol. 34, no. 10, pp. 1537-1557, Oct. 2015.

[22] C. D. Schuman, "The effect of biologically-inspired mechanisms in spiking neural networks for neuromorphic implementation," *2017 International Joint Conference on Neural Networks (IJCNN)*, Anchorage, AK, 2017, pp. 2636-2643.

[23] P. A. Merolla *et al.,* "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668-673, Aug. 2014.

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

[25] M. W. McCurdy, M. H. Mendenhall, R. A. Reed, B. R. Rogers, R. A. Weller and R. D. Schrimpf, "Vanderbilt Pelletron - Low Energy Protons and Other Ions for Radiation Effects on Electronics," *2015 IEEE Radiation Effects Data Workshop (REDW)*, Boston, MA, 2015, pp. 146-151.

[26] J. F. Ziegler, SRIM, 2013 [Online]. Available: http://www.srim.org.

[27] D. Decoste and B. Schölkopf, "Training Invariant Support Vector Machines," *Machine Learning*, vol. 46, pp. 161-190, 2002.

[28] F. Chollet *et. al.*, Keras, 2015 [Online]. Available: http://keras.io.

[29] M. Abadi *et. al.,* TensorFlow, 2015 [Online]. Available: http://tensorflow.org.