Optimized Hyperdimensional Edge AI Evaluation for Efficiency and Reliability under Real Radiation

ANUJ JUSTUS RAJAPPA, University of Antwerp - imec, IDLab - Faculty of Applied Engineering, Belgium
LAURA SMETS and PHILIPPE REITER, University of Antwerp - imec, IDLab - Department of Computer Science, Belgium
PAOLO RECH, Università di Trento, Italy
YNTE VANDERHOYDONC, RITESH KUMAR SINGH, and SIEGFRIED MERCELIS, University of Antwerp - imec, IDLab - Faculty of Applied Engineering, Belgium
JEROEN FAMAEY, University of Antwerp - imec, IDLab - Department of Computer Science, Belgium

Hyperdimensional Computing (HDC) is an emerging AI algorithm, touted to be an efficient, neuro-inspired and reliable alternative to neural networks for Edge AI. HDC utilizes hypervectors with several thousand elements; the number of elements in these hypervectors denotes the HDC dimension. This dimension can be optimized for improving the efficiency and reliability of HDC inference against errors such as bit-flips, which can be caused by environmental radiation-induced soft errors. We hypothesize that, by reducing the runtime chip area and execution time utilized by HDC inference through lowering dimensionality, both efficiency and reliability against soft error-induced bit-flips can be simultaneously improved while trading off a negligible amount of accuracy and error threshold. We tested our hypothesis by executing an HDC inference algorithm with two different dimension values, 10000 (10k) and 1024, on a commercially available, low-power, bare-metal ARM platform with a Cortex-M4 processor. We conducted the efficiency analysis by measuring the CPU cycles and energy required for executing the algorithm, and the reliability analysis using real-world atmospheric-like neutron radiation from the ChipIr facility in Oxfordshire, UK. Analyses revealed that, by lowering the HDC dimension from 10k to 1024, the reliability of HDC inference against soft error-induced bit-flips was 3.5 times better and efficiency improved by more than 16 times. This innovative observation contrasts the prevailing understanding in the community that increasing the HDC dimension always improves robustness or reliability. To the best of our knowledge, our work is the first to study the reliability of HDC inference using real-world radiation.

CCS Concepts: • **Computer systems organization** → **Reliability**; • **Computing methodologies** → *Bio-inspired approaches*; • **Hardware** → **Fault tolerance**; *Power estimation and optimization.*

Additional Key Words and Phrases: AI, COTS, hyperdimensional , low-power, radiation, robustness, single-event upset, soft errors

Authors' addresses: Anuj Justus Rajappa, anuj.justusrajappa@uantwerpen.be, University of Antwerp - imec, and IDLab - Faculty of Applied Engineering, Sint-Pietersvliet 7, Antwerp, Belgium, 2000; Laura Smets, laura.smets@uantwerpen.be; Philippe Reiter, philippe.reiter@uantwerpen.be, University of Antwerp - imec, and IDLab - Department of Computer Science, Sint-Pietersvliet 7, Antwerp, Belgium, 2000; Paolo Rech, paolo.rech@unitn.it, Università di Trento, Via Calepina, 14, Trento, Italy, 38122; Ynte Vanderhoydonc, ynte.vanderhoydonc@uantwerpen.be; Ritesh Kumar Singh, riteshkumar.singh@uantwerpen.be; Siegfried Mercelis, siegfried.mercelis@uantwerpen.be, University of Antwerp - imec, and IDLab - Faculty of Applied Engineering, Sint-Pietersvliet 7, Antwerp, Belgium, 2000; Jeroen Famaey, jeroen.famaey@uantwerpen.be, University of Antwerp - imec, and IDLab - Department of Computer Science, Sint-Pietersvliet 7, Antwerp, Belgium, 2000.

Manuscript submitted to ACM

ACM Reference Format:
Anuj Justus Rajappa, Laura Smets, Philippe Reiter, Paolo Rech, Ynte Vanderhoydonc, Ritesh Kumar Singh, Siegfried Mercelis, and Jeroen Famaey. 2024. Optimized Hyperdimensional Edge AI Evaluation for Efficiency and Reliability under Real Radiation. 1, 1 (November 2024), 24 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

As a rapidly emerging, neuro-inspired aritifial intelligence (AI) algorithm of interest, Hyperdimensional Computing (HDC) is positioned to be a superior alternative to neural network (NN) in terms of reliable and resource-efficient execution, making it a suitable candidate for Edge AI [7, 12]. HDC algorithms inherently support parallel processing and utilize bitwise operations rather than the intensive multiply-accumulate operations found in NN and deep learning. Hence, several hardware implementations have been developed to leverage this efficient execution of HDC [12]. However, any changes to hardware are often costly to implement [57] or may not be realizable with existing architectures. Hence, we focused on software changes for HDC algorithms and evaluated those changes using an existing hardware platform.

Bit-flips, where a bit becomes inadvertently inverted from zero to one or one to zero, can be caused by several factors, such as cosmic and atmospheric radiation, radioisotopic impurities in the packaging and chip materials [11], voltage scaling [27], readout errors in memories [38, 57], temperature fluctuations, aging and technology scaling [59, 60]. HDC uses hypervectors (HVs); i.e., vectors made of thousands of elements that leverage holographic representation, where each element is made of single or multiple bits. Hence, when bits or elements fail, the information degrades in relation to the number of failing elements irrespective of their position. Therefore, HVs are inherently more reliable against bit-flips when compared to position-dependent standard binary representations where every bit counts [26].

However, understanding the overall reliability of HDC against hardware errors such as bit-flips is key to employing it in mission-critical domains. Hence, there is a need for all possible hardware components in an HDC model to be considered for a comprehensive evaluation of its reliability against hardware errors as most studies do not consider all parts of an HDC model in their evaluation [38]. Besides, the efficiency of HDC systems should also be considered during its design [38]. These two considerations have been addressed in this work.

1.1 Soft error-induced bit-flips

Electronic devices in places such as office buildings, nuclear reactors or outer space are constantly being bombarded by energetic sub-atomic particles such as neutrons, protons, electrons and ions that are usually not visible to the naked eye. Some of these particles are capable of ionizing atoms (knocking an electron off an atom) either directly or indirectly.

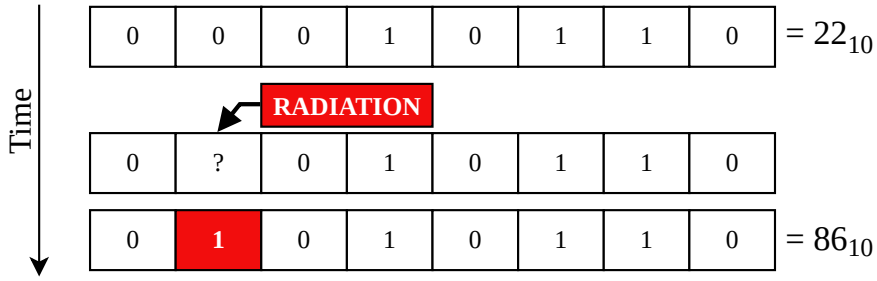


Fig. 1. A bit flip due to radiation in an 8-bit memory

The energy emitted in the form of these moving ionizing particles is called radiation [23]. This article does not consider the energy emitted in the form of electromagnetic waves, which is also known as radiation [23].

This particle-based radiation can sometimes interact with semiconductors present in electronic devices, which can cause a variety of malfunctions [19, 36]. One such malfunction shown in Figure 1 is a single bit-flip, where a bit of data can inadvertently change from representing 0 to 1 or 1 to 0. This affects the reliability of the data stored in the memory. For instance, Figure 1 shows a single bit-flip deviating the value 22 stored in an 8-bit memory and increasing it by almost 4 times, with room for more deviation! This type of error is called a soft error as it can be corrected by performing certain normal functions, such as a write, which is in contrast to hard errors that are not correctable [23].

Multiple terms are used to quantitatively analyze the effect of radiation due to a given type of particle. For instance, within a given period of time, the number of particles incident on a surface divided by the surface area is referred to as fluence [23]. Within that given period of time, the number of soft errors encountered by the device under the surface area divided by fluence is referred to as soft error cross-section, which is a measure of the device's susceptibility to soft errors for that type of particle and it depends on the device's operating conditions [23]. The time rate of change of fluence is called flux [23]. The articles [19] and [36] have detailed information on the effects of radiation, particle accumulation and their propagation. The articles [43] and [42] analyze the signals involved in radiation effects such as clock and data. Various terms and knowledge about the radiation experiment are detailed in Section 2, Section 6 and Annex A of the JEDEC JESD89B:2021 standard document [23].

These bit-flips can lower the reliability of HDC inference, a process that is paramount to safety-critical domains using Edge AI, such as autonomous vehicles, medical instruments [40], avionics [29], Internet of Space Things (IoST) [44], and nuclear power plants [48]. Due to the abundance and impact of neutron radiation in real-world terrestrial atmospheric conditions [23], we studied the impact of soft error-induced bit-flips [23] on HDC inference using atmospheric-like neutron particle-based radiation from the ChipIr facility in Oxfordshire, UK [11, 13] with a low-power, bare-metal commercially available off-the-shelf (COTS) test device, whose results are also applicable in space environments [10, 14].

1.2 HDC algorithm

The two main processes associated with the HDC algorithm executed for the study are the *inference* with the test device and the *training* on a personal computer (PC), using the following hyperdimensional arithmetic vector operations [53].

Elementwise XOR: When two or more HVs with binarized elements (binary HV) are considered, they can be combined using the XOR boolean operation. Here, the elements with the same index in the considered HVs undergo XOR operation, and the output from XOR is stored in the same index of an output HV, thus, the output is a binary HV.

Elementwise addition: When two or more HVs with integer elements (integer HV) or binarized elements are considered, they can be combined using the addition operation. Here, the elements with the same index in the considered HVs undergo addition, and the addition output is stored in the same index of an output HV, thus, the output HV has integer elements. The maximum value an element can reach in the HV (MAX_e) of the output when only binary HVs are considered is the number of HVs considered, thus, each binary HV adds +1 towards MAX_e of the output HV. When integer HVs are also considered, they add their $+MAX_e$ value towards the output HV's MAX_e, instead of +1.

Elementwise subtraction: When two or more binary or integer HVs are considered, they can be combined using the subtraction operation. Here, the elements with the same index in the considered HVs undergo subtraction, and the subtraction output is stored in the same index of an output HV, thus, the output HV can have integer elements. Elementwise subtracting a binary HV from an integer HV with MAX_e value is the only scenario considered in this article, and $MAX_e - 1$ is the new MAX_e of the output HV. This article does not consider other subtraction scenarios between

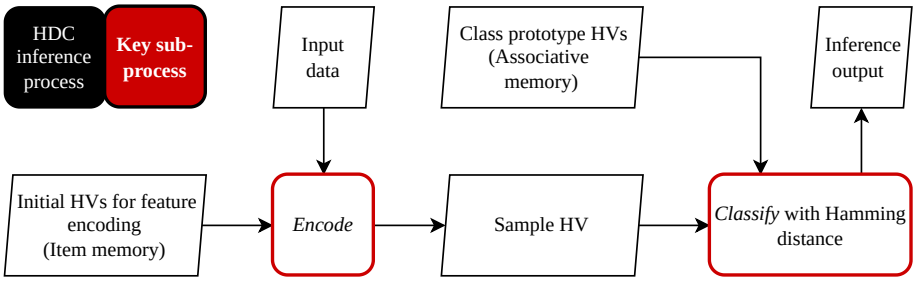


Fig. 2. Inference process of the HDC algorithm executed on the test device

integer or binary HVs. Should a subtraction cause the MAX_e of the output HV to become zero, the corresponding subtraction operation is aborted.

Majority rule: To binarize an integer HV with a MAX_e value, any element in the integer HV whose value is less than $(MAX_e/2)$ is zeroed and any element greater than that becomes one. But, if an element's value is equal to $(MAX_e/2)$, then either zero or one is chosen at random to replace it. The resulting binary HV is the output of the majority rule.

Note that the above operations do not change the dimensionality of the HVs. The elemental XOR operation is also known as **binding**. When an HV is **bundled with** another HV, those HVs undergo elementwise addition followed by the majority rule. When an HV is **bundled out** of another HV, those HVs undergo elementwise subtraction followed by the majority rule. These binding and bundling operations are used by the inference process shown in Figure 2, and the training process shown in Figure 3. Both processes start by accessing the initial set of HVs used to generate a sample hypervector (HV) by encoding the input data (or input image). While training, this sample HV and the input data's label are used for building class prototype HVs. Whenever inference occurs with input data, these class prototype HVs built from the training process and the sample HV of the input data are used to generate the inference output. The key subprocesses in an HDC inference are the Encode and Classify blocks, and the key subprocesses in HDC training are the Encode, Build class prototype and Classify blocks, which are marked with a red outline in Figure 2 and Figure 3. These subprocesses are further explained in the following subsections.

1.2.1 Encode. First, input data that is a greyscale image with the pixel values $[0, 255]$ is obtained. Second, the encoding involves binarizing all pixel values of the input data. The binarization process involves changing the pixel value to 1 if it is greater than zero. Each pixel has an HV representing it (HV_{pix}). The HV_{pix} for the pixel with value 0 is represented

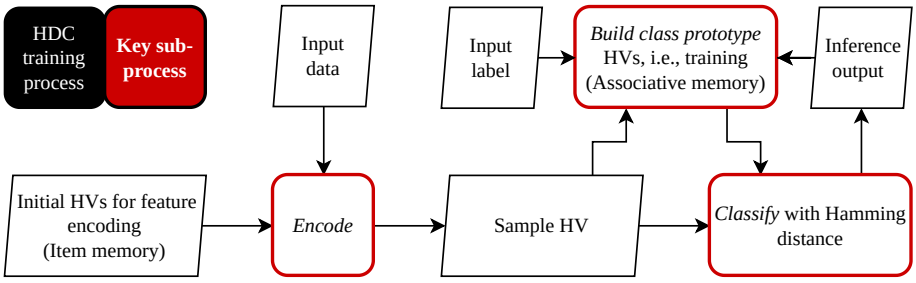


Fig. 3. Training process of the HDC algorithm executed on a PC

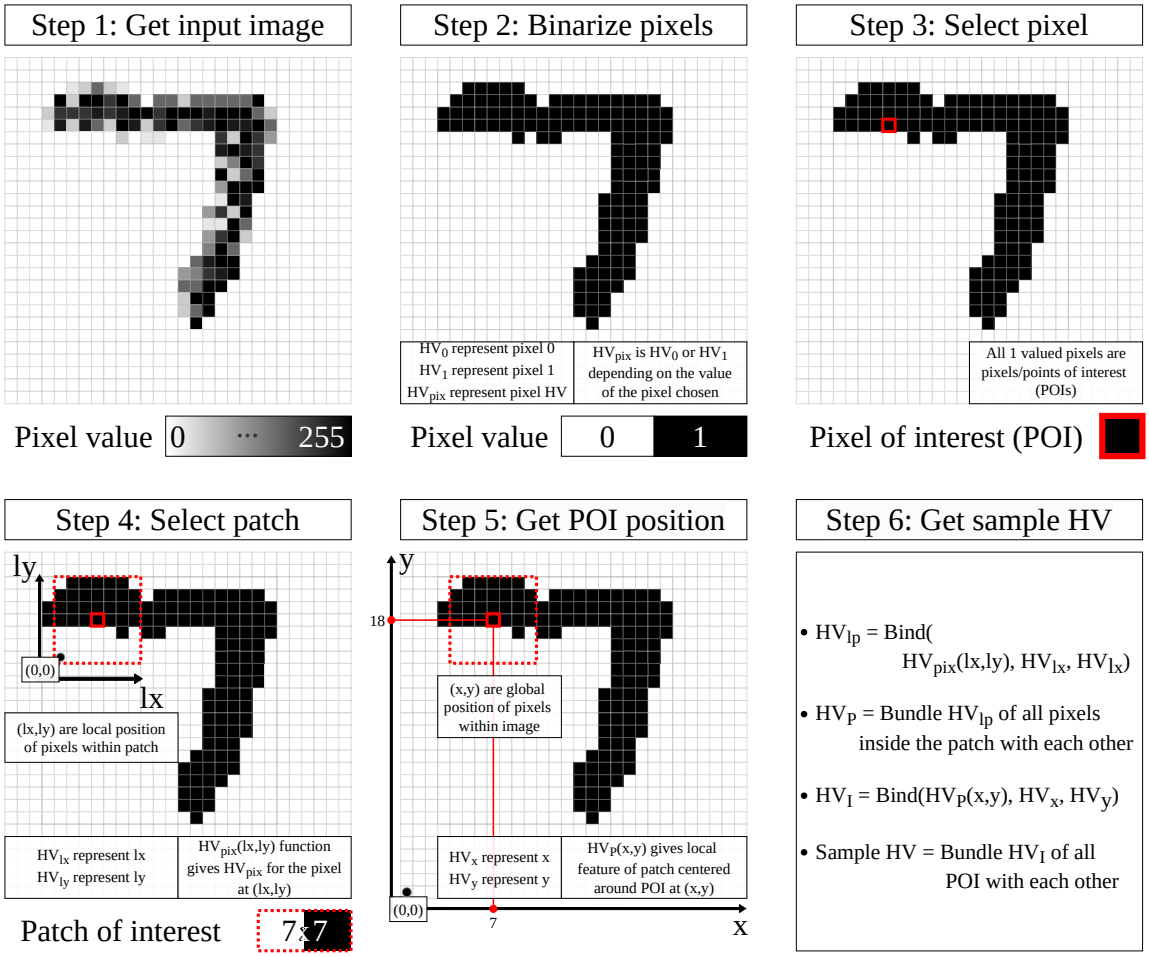


Fig. 4. Encode subprocess of the HDC algorithm using 28x28 pixel greyscale image as input data

by HV_0. The HV_{pix} for the pixel with value 1 is represented by HV_1. Third, the pixels (or points) of interest (POI) are selected from within the binarized image pixels. Here, we chose pixels with value 1 as POI [53].

Fourth, the binary value of that POI and its neighbouring pixels are transformed into a patch HV (HV_P) to get the local feature of that POI. In our case, the neighbouring pixels form a patch of 7x7 pixels, with POI at its center. Each pixel within this patch of interest is used to calculate HV_P. Let lx and ly be the position (coordinates) of these pixels within the patch. Then the HV of these pixels is given by the function $HV_{pix}(lx, ly)$. Each lx has an HV (HV_{lx}) to represent it and each ly has its own HV (HV_{ly}) representing it. Binding $HV_{pix}(lx, ly), HV_{lx}, HV_{ly}$ provides HV_{lp}, which is the feature of the pixel at (lx, ly) within the patch. The HV_{lp} of all the pixels within the patch of interest are bundled with each other to form HV_P, which is the local feature associated with the POI.

Fifth, the x and y representing the position (coordinates) of the POI within the entire input image (global position) is obtained. Each x has an HV (HV_x) to represent it and each y has its own HV (HV_y) representing it.

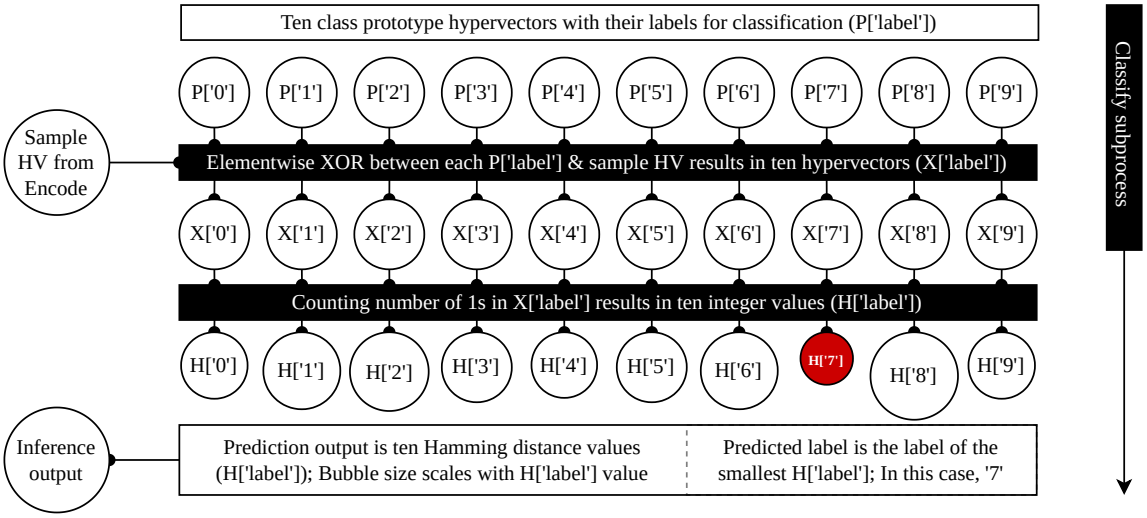


Fig. 5. Classify subprocess of the HDC algorithm for ten-digit (MNIST) classification

The HV_P undergoes one more transformation into image HV (HV_I) to factor in the global position of the POI. Let the HV_P for a POI at position (x,y) be given by the function $HV_P(x, y)$. Then HV_I is obtained by binding the $HV_P(x, y)$,HV_x and HV_y. The HV_I of all the POIs are bundled with each other to generate a sample HV, that represents the entire input image. Obtaining sample HV marks the end of encoding input data. The steps associated with the encode subprocess are also shown in Figure 4. The algorithm uses native hyperdimensional arithmetic vector operations [53] end to end, without relying on other algorithms such as neural networks. The transformations above use a pseudo-randomly generated initial set of HVs, which are $HV_0, HV_1, HV_{lx}, HV_{ly}, HV_x,$ *and* HV_y, and remain constant throughout the model's lifetime. The memory containing this initial set of HVs is called item memory [54]. This encoding subprocess is the proposed framework in [53], which can be conferred for more details.

1.2.2 Classify. Classify uses binarized class prototype HVs, obtained from the Build class prototype subprocess, for HDC inference. Classify involves obtaining the sample HV from the Encode subprocess and calculating the Hamming distance between this sample HV and each of the class prototype HVs. The pair with the lowest Hamming distance is said to be highly similar, and the class of the class prototype HV in that pair is the **predicted class or label**. The set of Hamming distances calculated between these pairs forms the **prediction output**, which, with the predicted label, forms the **inference output**. Hamming distance between two binary HVs is calculated by applying elementwise XOR operation on them and counting the number of 1s in the output HV. For more information on operations such as elementwise XOR and counting 1s, please refer to Section 2 of [54]. The classify subprocess for identifying the class or label associated with the encoded input data (sample HV) among ten classes is shown in Figure 5.

1.2.3 Build class prototype. Class prototype HVs, also known as class HVs are generated from the input labels, sample HVs and inference output available during training. Datasets for training contain both the input data and the corresponding label or class assigned to that input. Each class has a corresponding class prototype HV. They are initialized by bundling all the sample HVs belonging to the corresponding class with each other. While training, each sample HV generated from the MNIST training dataset once again gets bundled with the corresponding class prototype

HV if the input label disagrees with the predicted label. Meanwhile, this sample HV is also bundled out of the class prototype HV corresponding to the predicted label. But, when the prediction label agrees with the input label, no operation takes place. This process can be iterated for multiple epochs until the desired model accuracy is reached. After each epoch, the class prototype HVs are binarized using the majority rule and used for inference during the next epoch. The memory containing these binarized class prototype HVs is called associative memory [54]. For more information, confer Section 2 of [54].

1.3 Hypothesis

We hypothesize that, for a given neutron spectrum [23] , decreasing the runtime chip area and execution time used by the HDC algorithm through reducing its dimensionality will decrease the chances of a neutron-induced bit-flip impacting the inference output. This would thereby improve the reliability and efficiency of the inference. Arguments partially supporting our hypothesis can be found in the NN related works [15, 47], which also associate runtime chip area and execution time, later defined as vulnerable area-time factor (VAT) in this work, to reliability. On another note, a previous work [39], which studied dimensionality reduction in terms of accuracy and input data distortion using computer simulations, found the accuracy of lower dimensional HDC to be maintained closer to that of the higher dimensionalities and recommended dimensionality reduction for hardware implementation of HDC due to the associated beneficial chip area and power reductions. By contrast, we implement dimensionality reduction through software changes and verify its impact on reliability and efficiency using real-world experiments on a low-power, bare-metal COTS device. Thus, this previous finding and our work are complementary.

The intuition behind our hypothesis can be obtained by imagining a fishnet to catch fish. The bigger the fishnet cast and the wait time, the more fish could be caught. Here, the fish represents a neutron, the fishnet represents the runtime chip area, and the wait time represents the execution time. The key difference between this scenario and our hypothesis is that '*catching fish is usually favoured while catching neutrons is not!*' So we reduce the runtime chip area and execution time as much as possible.

While the following argument could be applied to any radiation particle that is capable of inducing soft errors when it strikes, we will consider neutrons and that a single such particle can create soft errors, independent of other such particles [36]. Consider an HDC inference executing in our test device, consuming an average runtime chip area of a_r during its execution time t_e. Let the probability of a neutron impacting the HDC inference by interacting with the chip area a_r within time t_e be E_n. Then, the probability of the same neutron not impacting the inference is $\bar{E}_n = 1 - E_n$. If N is the total number of neutrons to pass through a_r within t_e, then the probability of none of those neutrons impacting the inference is $(\bar{E}_n)^N$, while the probability of at least one of those neutrons impacting the inference PE_n is the probability complement of none of those neutrons impacting the inference; i.e., $PE_n = 1 - (\bar{E}_n)^N$. Thus, reducing this probability, PE_n, is beneficial. Here, we assume $\bar{E}_n, N \in \mathfrak{R}$; $0 < \bar{E}_n < 1$; and $N > 0$ for all practical purposes. Hence, PE_n scales with N as increasing N decreases $(\bar{E}_n)^N$, causing its complement PE_n to increase.

For a given neutron spectrum, let ϕ_n be the overall constant neutron flux [23] in the environment containing the test device, which is the number of neutrons that pass through a unit area within a unit of time. Hence, multiplying flux ϕ_n by chip area a_r and time t_e gives us the number of neutrons of interest; i.e., $N = \phi_n \times a_r \times t_e$, where we assume $\phi_n, a_r, t_e \in \mathfrak{R}$ and $\phi_n, a_r, t_e > 0$ for all practical purposes. Let the VAT, theoretically denoting the vulnerability of HDC inference to chip area a_r and time t_e be given as $a_r \times t_e$. Then, $N = \phi_n \times VAT$. Therefore, decreasing VAT would decrease N, consequently scaling down PE_n, which is beneficial. We propose reducing both a_r and t_e via dimension

reduction in the HDC algorithm to reduce VAT. We expect the reduction in VAT to decrease the susceptibility of HDC inference to soft errors, which is experimentally observed as a reduction in soft error cross-section.

The surface area of the chip to which the HDC inference algorithm is vulnerable and exposed during the execution time t_e is not a constant as the data associated with the algorithm gets processed by different parts of the chip during execution. Thus, the vulnerable runtime surface area associated with the HDC inference changes with time and can be represented as $a(t)$, which is a function of time t. Then $VAT = \int_{t_1}^{t_2} a(t)\,dt$, where t_1 is inference execution start time, t_2 is inference execution end time and $t_2 - t_1 = t_e$. However, obtaining $a(t)$ directly requires extensive analysis of the proprietary parts and implementation details of the exact test device, which are typically unknown to application designer [36] and may not apply to other types of devices executing the same inference. Hence, we made a reasonable assumption that the HDC algorithm's total memory consumption is directly proportional to a_r and used this assumption to show the reduction of a_r due to dimension reduction in HDC. This assumption is applicable for a variety of COTS devices, as storing more bytes in the memory utilizes more circuitry on the chip [17].

We experimentally verified our hypothesis by executing the HDC inference algorithm on the bare-metal test device using a single process CPU core inside the radiation chamber at the ChipIr facility, including all components of HDC inference model in our real-world radiation experiment. The facility uses a neutron beam [11, 13] with the spectrum like the one referenced in JEDEC JESD89B:2021 standard document [23]. We measured the CPU cycles and energy required for executing the entire HDC inference algorithm to perform efficiency analysis. At least two different HDC dimensions are required to evaluate our hypothesis. We selected 10000 (10k), which is widely used among the HDC community, and 1024, due to its prevalent use in computing standards, including the RISC-V [49] architecture and Intel's vector extensions [22]. The evaluations were limited to two values for the HDC dimensions due to cost and time constraints for neutron beam time usage at ChipIr.

2 STATE-OF-THE-ART

Multiple studies [38] have been conducted to analyze the reliability of HDC against bit-flips using Simulated Fault Injection (SFI) [51] as the error source. This is often limited to the associative memory [54] and might not include item memory [54] or other compute units such as the registers and circuits used during the HDC computations [38].

For instance, [5, 60] uses SFI to assess the robustness of HDC models to associative memory errors. They also propose error-masking schemes, which involve zeroing the susceptible bits. This requires error detection techniques to identify the bit-flips, which can add additional overhead. Others [27, 34, 45, 46, 57] use SFI to simulate memory failures and assess their impact on their custom hardware design for HDC, whose results might not be compatible with COTS devices due to differences in the hardware designs.

DependableHD [33] is a training framework to improve reliability against bit-flips. They use SFI in their simulated custom hardware for assessment. They propose the margin enhancement technique for risky predictions and introduce random noise during training for reliability improvement. DependableHDv2 [34] adds the dimension-swapping technique to DependableHD for handling stuck-at errors in memory by swapping the elements in all the base and class HVs post-training and may not be suitable for all types of HDC models. HyperMetric [57] is another framework that introduces their approximate encoding technique and updates both the encoder weights and class HVs during training. They showed HyperMetric to perform better than their baseline with voltage-scaled, SRAM-induced bit-flips in their simulated hardware implementation using SFI.

ScaleHD [59] is a software technique applied once to the HDC model post-training, assessed using SFI. It scales the values of the elements in a HV to reduce the relative error due to a bit-flip in a given position. However, it may not be

applicable if the elements in a HV are binary. StocHD [45] is an end-to-end system for HDC that includes learning over raw data without expensive pre-processing. It mathematically defines stochastic arithmetic over HVs and uses HDC data representation throughout its solution. They subjected their models with two different dimensions to the same levels of memory errors. Their results suggest that a higher dimension translates to lower quality loss and, thus, higher reliability against memory errors.

A study focusing on the effect of precision of the elements in HVs and the dimension of HDC on robustness suggests using vectors with lower precision and higher dimensional representation for more reliability against bit-flips [18]. While [60] also suggests using lower precision, such as 1-bit (binary), they found the lower dimensional representation more reliable against bit-flips. This contrast in observation prompts the need for real-world experimental results to validate simulations, which, to the best of our knowledge, we are the first to provide in this work with particle radiation, in the form of neutron beams, as the source of these bit-flips in HDC inference.

The above works do not directly consider the runtime chip area and execution time in their reliability against bit-flips analysis, which are important factors when particle radiation is the source of bit-flips. In our radiation experiment, these factors are considered via placing the whole test device executing HDC inference inside the radiation chamber. Thus, our key contributions to the state-of-the-art through this work include the following: introducing dimension reduction as a software optimization to improve HDC's efficiency and reliability against soft error-induced bit-flips in COTS devices; introducing and theoretically analysing our hypothesis; experimentally verifying our hypothesis by executing the HDC algorithm in a low-power, bare-metal COTS test device under the atmospheric-like neutrons in the radiation chamber at the ChipIr facility in the UK, analyzing the output from the radiation experiment, and measuring the execution energy and execution time of the HDC algorithm using the test device; and finally, publishing the data associated with this work [25]. More information about the methods involved in the execution time and execution energy measurements for efficiency analysis and the radiation experiment are in the next section, the results from the measurements for efficiency analysis and the radiation experiments for reliability against soft error-induced bit-flips analysis along with their implications are in Section 4, and the conclusion is in Section 5.

3 METHODOLOGY

This section contains information about the software and hardware setup used for the measurements and experiments. The results of those measurements and experiments are presented and discussed in the next section. We use two versions of HDC algorithm (case study versions), one with 10k dimension (**HDC10k** version) as a baseline, whose inferences are more probable to be impacted by soft error-induced bit-flips than the other with 1024 dimension (**HDC1024** version), due to the difference in their execution time and memory consumption (runtime chip area). Apart from the dimensional change, both case study versions are the same and were designed and trained using C-language. The HVs are stored using 32-bit half-words in memory. Each HV in HDC10k uses 313 half-words, while HDC1024 uses 32 half-words. The bits in these half-words are processed one bit at a time using bit-wise operations. As an HV of the HDC10k does not perfectly fit within 313 half-words, the remaining 16 bits of the last half-words are not considered and no processing on these bits is performed.

We used the MNIST [31] database for our analysis to be in line with several other HDC related works [7, 18, 27, 34, 38, 39, 57, 59]. The case study versions were trained using the MNIST training dataset and their accuracy was evaluated on a PC outside the radiation chamber. The model accuracy was calculated with the entire MNIST test dataset for both the case study versions. The HVs from training are then compiled with the C-implementation of the case study versions to generate HDC inference executables for the test device with the Segger compiler [3]. To manage the

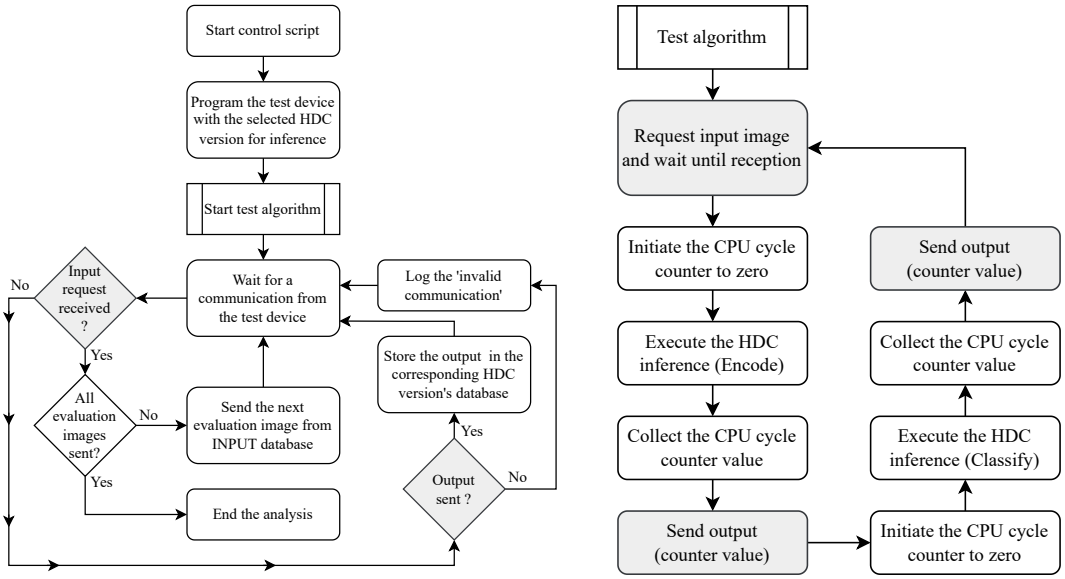
radiation experiments within the provided beam time constraints (cf., Section 4.2), 10 images from each of the 10 classes were randomly selected from the MNIST test database and used for evaluating both case study versions. These 100 evaluation images were used as inference input throughout the following experiments and analyses. The evaluation images were stored in the INPUT database of MongoDB using a Python script.

The low-power, bare-metal COTS test device used for executing the case study versions is the Nordic nRF52840 DK [2], which contains a single ARM-based Cortex-M4 processor operating at 64 MHz without an operating system (OS). The efficiency of inference with both case study versions was analyzed outside the radiation chamber by measuring their execution time and energy consumption in this test device. The process or execution flows used by the programs running in the test device are implemented as a single software process/thread, where the instructions are executed consecutively.

3.1 Execution time measurement

The execution time required for executing an HDC inference was measured in CPU cycles. The number of CPU cycles required for inference was measured using the Data Watchpoint and Trace Unit (DWT) [1] available in the test device. It consists of a 32-bit register whose value increments for each CPU cycle when enabled. This register can overflow when measuring events that progress beyond a certain time period. Hence, the approximate start and end timestamps of such events are collected and combined with the value of this register for a more accurate CPU cycle count of such longer executing events. The time in CPU cycles when divided by the CPU frequency value gives the time in seconds.

To automate the collection of execution time data from the test device for both the case study versions using all the evaluation images, a control script was developed with the Python language. The case study version of interest is selected via a configuration file, which the control script uses to program the test device after it is started manually



(a) Program flow used by the control script executed on a PC.

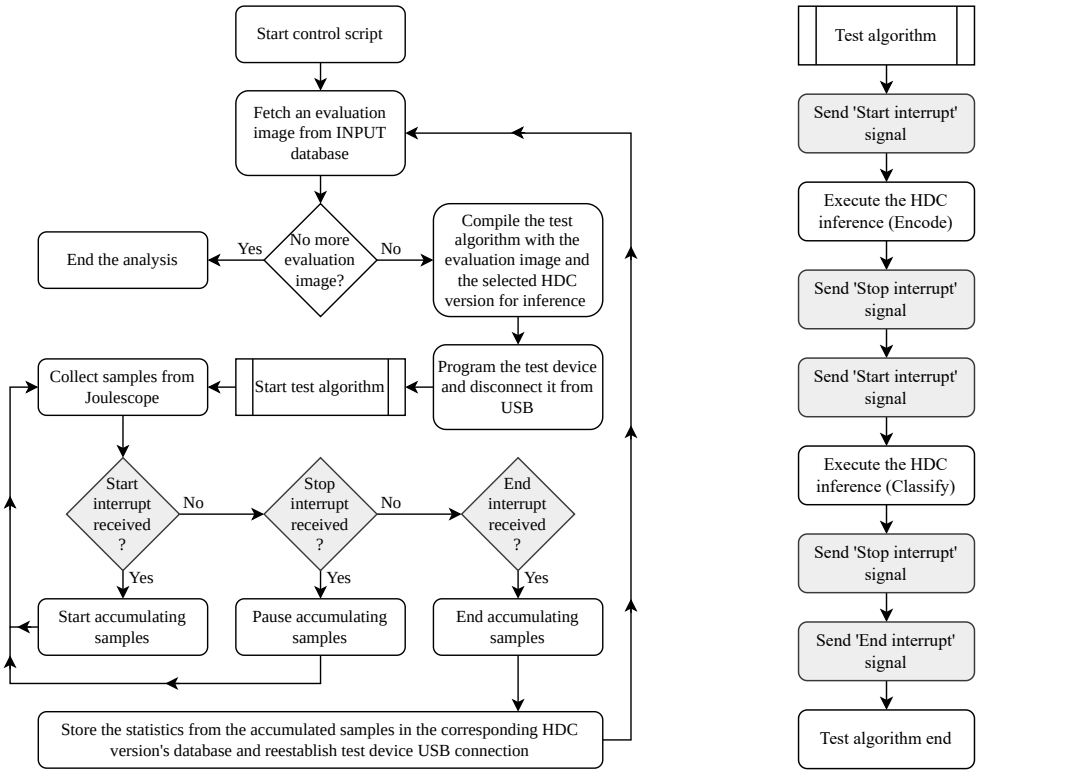
(b) Program flow used by the test device.

Fig. 6. Program flows used for the execution time measurement.

on a PC. Once the test device is programmed, the test device is reset to start the programmed test algorithm. The test algorithm requests an input image for HDC inference. The control script receives this request and sends an image from the evaluation images. When all the evaluation images are sent, the analysis is complete. For each HDC inference, the test device sends two CPU cycle values, one for the Encode process and another for the Classify process (cf., Section 1.2). They are added together to get the total execution time of the HDC inference process. The program flow of the control script is shown in Figure 6a and the corresponding test algorithm running on the test device is shown in Figure 6b.

3.2 Execution energy measurement

The power supplied to the test device was monitored using a Joulescope [24] while executing the inference algorithm for energy measurement in Joules. Joulescope samples both the current and voltage over time to provide the energy consumption values. The external DC power supply, a Multicomp pro MP710078 [41], was used to power the test device via the Joulescope during the energy measurements while keeping the input voltage fixed at 3 V. A USB interrupter module was used to interrupt the flow of power through the direct USB connection between the test device and the PC conducting the analysis when needed, as the USB power can interfere with the energy measurements. The PC controls the Joulescope and the test device during the execution energy analysis through USB connections. The PC also controls



(a) Program flow used by the control script executed on a PC.

(b) Program flow used by the test device.

Fig. 7. Program flows used for the execution energy measurement.

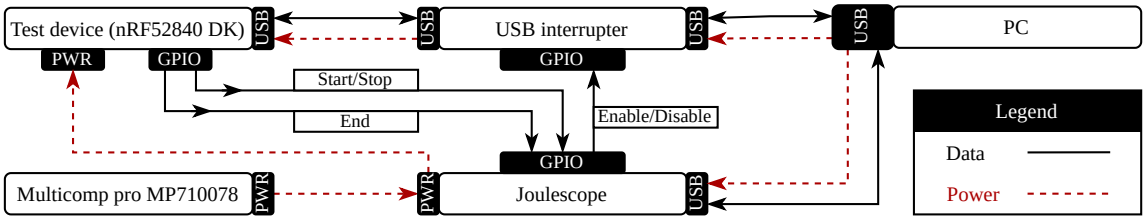


Fig. 8. Hardware setup used for execution energy measurement.

the USB interrupter module via the general-purpose I/O (GPIO) pins available in the Joulescope. The entire hardware setup for measuring the execution energy of HDC inference is shown in Figure 8.

A control script was written using the Python language with the program flow shown in Figure 7a and it is used to automate the execution energy data collection for each of the case study versions. The script works with the test algorithm, whose program flow is shown in Figure 7b. The test algorithm runs on the test device to measure the energy consumed by the Encode and Classify subprocesses of an HDC inference, which are combined to provide the overall energy consumption of an HDC inference. Once the case study version to analyse is selected, it is entered into a configuration file that the control script reads when it is manually started. Then, it fetches an evaluation image from the INPUT database and compiles it with the selected case study version to generate a test algorithm. The test device is programmed with the compiled executable. The direct USB connection between the PC and the test device is disconnected by enabling the USB interrupter via the Joulescope, and the test device resets to start the test algorithm. Now, the samples from the Joulescopes are recorded continuously, which includes the energy measurements and the status of its GPIOs. The test device sends the Start and Stop interrupts to a GPIO before and after the Encode and Classify subprocesses of the HDC inference. Here, the Start interrupt is a falling edge interrupt and the Stop interrupt is a rising edge interrupt. The samples are analysed and only the energy measurements between the Start and Stop interrupts are accumulated to calculate the energy consumed by the Encode and Classify subprocesses, together providing the total energy consumption of an HDC inference.

The Joulescope has a sampling rate of 2MHz. The latency of the test device, data lines and the associated sampling error are analyzed in another work by Huybrechts et al. [20]. From this work, it is established that the sampling error impact decreases with increasing task length, i.e., execution time. Hence, we consider the multi-second execution time for inferences compared to the microsecond sampling times to make sampling errors insignificant (cf., Section 3.4.3 in [20]). The analysis for the executable ends when the End interrupt (a rising edge) is detected on another dedicated GPIO of the Joulescope. The direct USB connection between the PC and the test device is reestablished by disabling the USB interrupter via the Joulescope. Once again a new image from the evaluation images is selected and the whole process repeats until no further evaluation images are left.

3.3 Radiation experiment

Once the test device is placed inside the radiation chamber, the executable and input images for inference are fed in via USB-ethernet connectors from a control server (PC) outside the radiation chamber. This is made possible by the USB-LAN extender system from ADI Icron [4], powered by a programmable power supply. The USB-LAN extender has a local and remote unit connected by ethernet through the facility's ethernet ports. Inside the radiation chamber, the

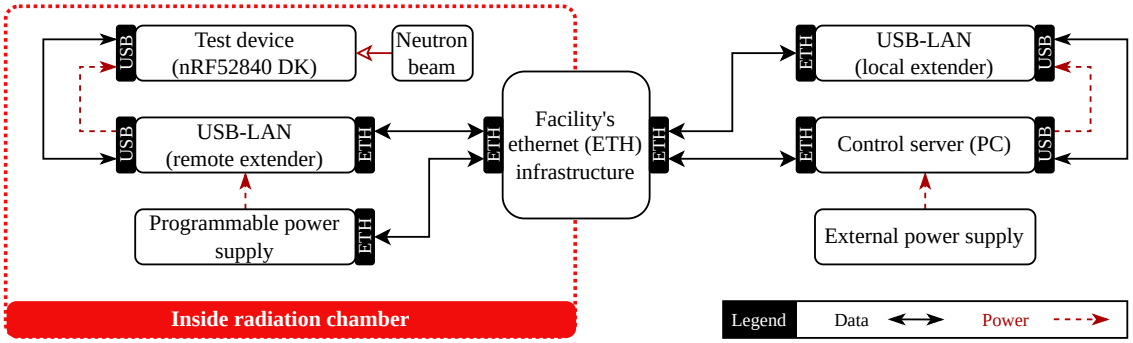
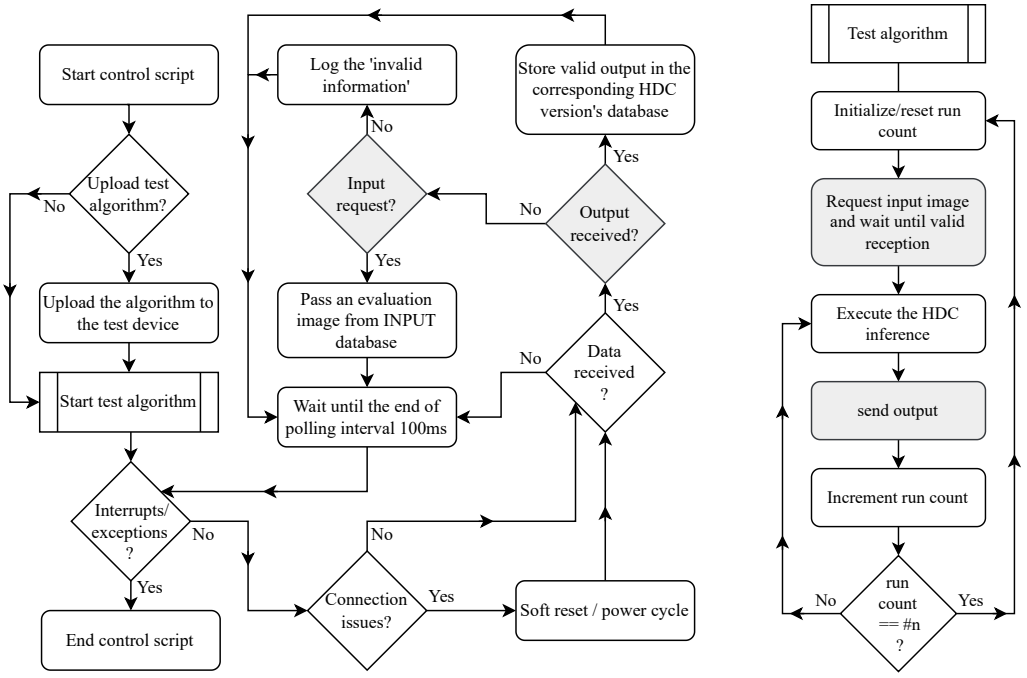


Fig. 9. System block diagram representing the radiation experiment setup.

USB-LAN remote extender powers the test device while remaining shielded from the neutron beam along with the programmable power supply and the facility's ethernet ports. The control server can control the test device via JTAG and power-cycle it when needed, while also collecting the experimental output from the test device and storing it in the corresponding database. Various processes running on the control server, associated with the radiation experiment, are automated using a control script written in Python. The experimental setup is depicted in Figure 9. The program flows used during the radiation experiment conducted at the ChipIr facility in the UK are shown in Figure 10.



(a) Program flow used by the control script executed on a PC.

(b) Program flow used by the test device.

Fig. 10. Program flows used by the radiation experiment at ChipIr

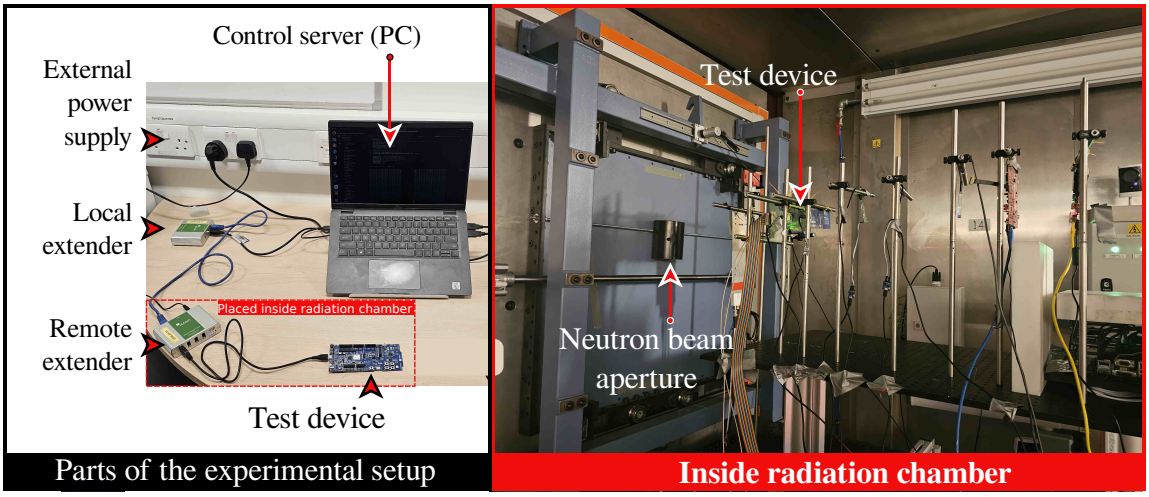


Fig. 11. Radiation experiment setup at ChipIr.

The control script and the corresponding test algorithm on the test device work in tandem during the radiation experiment. The program flow associated with the control script is shown in Figure 10a and the program flow associated with the test algorithm running on the test device is shown in Figure 10b. The control script reads the configuration file to: (1) select the executable of the test algorithm, (2) the database to store the experimental output, and (3) other such parameters for each case study version. Each executable of the test algorithm contains only one of the HDC versions. Thus, only one case study version is tested by the test device at a time. As it is not advisable to program the device while the neutron beam is active due to a fire hazard, user intervention is required by the control script to program the test device. Once the test device is ready, the test algorithm on the test device is started. The control script exits if any user interrupts or unhandled exceptions occur during the experiment. A soft reset through JTAG or power cycling via the programmable power supply is used to recover the test device from communication issues or unresponsive behaviour. If the test device is working as expected, then the control script checks for any data received from the test device. The real-world experimental setup at ChipIr is shown in Figure 11.

If the information in the data is a request for input, an evaluation image from the INPUT database is sent to the test device. Transferring the input image to the test device uses up experimental time that cannot be used for evaluating the HDC algorithm under radiation (i.e., useful beam time). To reduce neutron wastage, the minimum useful beam time is recommended to be 80% of the total beam time at ChipIr. To further reduce neutron wastage, we managed to keep the useful beam time at around 97% of the total beam time with both the case study versions. This was achieved by running one inference per transmitted input (i.e., $\#n = 1$) for the HDC10k version and 20 sequential inferences per transmitted input (i.e., $\#n = 20$) for the HDC1024 version, while also allowing us to transmit all the evaluation images within the allocated beam time.

If the information indicates that the device is sending output, then that is logged in the corresponding case study version's database. The test device sends output data after each inference as it is needed for analysis. This data includes the inference output (cf., Section 1.2.2) but does not include any HVs to reduce the output transfer time (communication overhead), which rendered this communication overhead insignificant. After sending all the output data, the test device

Table 1. $Average \pm Standard\ deviation$ statistics of resource consumed by HDC10k and HDC1024 versions in terms of execution time $(billion\ (or)\ giga\ (or)\ 10^9\ CPU\ cycles\ @\ 64\ MHz)$ and execution energy $(Joules)$, measured using all the evaluation images

Measurement	(Sub)Process	HDC10k version	HDC1024 version	Reduction
Execution Time (Giga-cycles)	Encode	7.20 ± 1.92	0.40 ± 0.10	94.4%
	Classify	3.80e-3 ± 1.68e-8	0.44e-3 ± 0.58e-8	88.4%
	Inference	7.20 ± 1.92	0.40 ± 0.10	94.4%
Execution Energy (Joules)	Encode	2.92 ± 0.74	0.18 ± 0.05	93.8%
	Classify	1.60e-3 ± 1.03e-6	0.21e-3 ± 9.33e-6	86.8%
	Inference	2.92 ± 0.74	0.18 ± 0.05	93.8%

sends another input request to the control script. If the information from data sent by the test device is invalid, it is logged. Finally, the control script waits until the end of the predefined polling period of 100 ms and repeats the process from checking for user interrupts.

4 RESULTS AND DISCUSSION

Any percentage of improvement/increase or reduction/decrease in a parameter is calculated using two values, the baseline value and the changed value, using the default formula $\frac{baseline\ value - changed\ value}{baseline\ value} \times 100\%$. In our case, when calculating the percentage change between HDC10k and HDC1024, the value associated with HDC10k is the baseline value and the one with HDC1024 is the changed value.

When changes are represented in $\#X$ times, the default formula used to calculate $\#X = \frac{changed\ value}{baseline\ value}$, where the changed and baseline values are similar to the ones described above. However, in the case of efficiency improvement changes, an increase in execution time, execution energy or memory consumption is associated with a decrease in efficiency. Similarly, the reliability improvement changes are considered inversely proportional to soft error cross-section and Failure In Time (FIT) values. Due to this inverse proportionality, $\#X = \frac{baseline\ value}{changed\ value}$ is used for $\#X$ times calculation.

We assume a_r to be directly proportional to the total memory consumption of the HDC inference executable ($a_r \propto memory\ consumption$), and use only the memory consumption to calculate the percentage change and $\#X$ times calculations of a_r and VAT (cf., Section 1.3). The compute area used by the HDC inference executable does not contribute to the a_r change, as it is considered static in our COTS test device. All the HVs used by the HDC inference are made up of half-words and undergo the same operations in both the case study versions. The percentage change and $\#X$ times for VAT are calculated using the value obtained by multiplying the memory consumption with the corresponding execution time for each case study version.

4.1 Efficiency analysis

Each HDC inference consists of two subprocesses Encode and Classify. The resources consumed by these subprocesses were measured during each inference and combined to get the resource consumed for that whole inference. Thus, we get the average and standard deviation statistics of resources consumed per Encode, Classify and inference in terms of time and energy. The various statistics obtained from the efficiency analysis for an HDC inference calculated using all the evaluation images are shown in Table 1. The data used for deriving the values shown in Table 1 can be found along with the published data [25].

As evident from the table, the resources consumed by Encode are orders of magnitude higher than Classify, contributing towards more than 99.9% of the resources consumed by an inference. Hence, further optimizing the encoding

Table 2. Radiation experiment results of HDC10k version and HDC1024 version

Measurements	HDC10k version	HDC1024 version
Unique error count	163	18
Irradiated inference count	308	2734
Irradiation time ($hours$)	9.95	4.78
Irradiated inferences per hour $(hours^{-1})$	31	572
Fluence $(10^{10} neutrons/cm^2)$	17.74	6.84
Soft error cross-section $(10^{-10} cm^2)$	9.18	2.63
Failure In Time (FIT)	11.88	3.40

scheme for efficiency can be highly beneficial. This can be done by leveraging hardware acceleration, replacing the encoding scheme with an equivalent efficient tiny NN model or using a different encoding scheme. Looking at the standard deviation values, the Encode subprocess deviates more from the mean than the Classify. This shows that the resources consumed by Classify remain relatively stable compared to the Encode subprocess between any two HDC inferences. It is also safe to conclude that the resources consumed by Encode are highly dependent on the input image used during an inference, as the evaluation image fed as input to the inference is different between any two inferences during the efficiency analysis. This could be primarily due to different input images having a different number of POIs and more POIs require more operations during Encode (cf., Subsection 1.2.1), affecting its efficiency.

Comparing the measurements of HDC10k with HDC1024 shows the efficiency benefits and low-energy nature of the dimension reduction optimization. Due to dimension reduction, the average execution time was reduced by 88.4% for the Classify subprocess and 94.4% for the Encode subprocess and inference process. The average execution energy was also reduced by 86.8% for the Classify subprocess and 93.8% for the Encode subprocess and inference process.

4.2 Experimental soft error-induced bit-flip analysis

The errors in the HDC inference during the radiation experiment are counted by comparing the experimental inference output with the golden inference output; i.e., the inference output obtained by executing the HDC algorithm outside the radiation chamber. It is considered an error if mismatches exist in these inference outputs for the same test image. Irrespective of the input data, the same errors in successive inferences executed within the radiation chamber are counted together as one **Unique error** and used for calculating soft error cross-section, as the probability of each of those inferences being affected by additional, distinct bit-flips or neutrons causing the same error in the HDC inference is assumed negligible. Errors in a series of successive inferences can often be remnants of a single bit-flip's impact on the first inference in that series. These errors can be mitigated using techniques such as scrubbing [35], which is out of the scope of this article.

As per Tables 1 and 2, the execution time of the HDC10k inference is about 18 times longer than that of HDC1024. Hence, the number of inferences executed (inference count) with HDC10k within a given time period will be about 18 times lower than what is possible with HDC1024. Therefore, the time during which both the case study versions were executed inside the radiation chamber was varied to achieve a high number of irradiated inferences and Unique error counts in both the versions for obtaining statistically significant results within the provided ChipIr beam time constraints. This includes setting the multiple of inferences per single input to one for HDC10k and twenty for HDC1024 (cf., Section 3.3). These results from the radiation experiment are shown in Table 2.

Beam time constraints: During the radiation experiment campaign, the neutron beams irradiating the test device were intermittently shut down as the beam source required repairs and maintenance at random intervals. This, combined with the total allocated access time to the ChipIr facility and manual interventions for pausing the neutron beam and reprogramming the device, which further required consent from other beam users at the facility as it also impacted their experiments, added to the constraints. As such, these constraints reduced the total available beam time and prevented testing multiple and more time-consumingly complex HDC algorithms using the test device.

Due to these constraints, not all the inferences executed by the test device inside the radiation chamber were irradiated. The irradiated inferences are identified and the irradiation time for the case study versions is calculated by analyzing the timestamps of the experimental output with the timestamps provided in the neutron logs made available after the end of the radiation experiments.

The neutron logs are also used to calculate the Fluence [23], which divides the Unique error count to provide the soft error cross-section values [23] for a given case study version. The soft error cross-section values denote the vulnerability of the test setup (i.e., the test device and the test algorithm) to the atmospheric-like neutron particles at ChipIr. It can be seen from Table 2 that vulnerability for HDC1024 is 71.4% lower than the HDC10k, based on their soft error cross-section values. This lower vulnerability is obtained for negligible tradeoffs discussed in Section 4.4.

The soft error cross-section values can be used to calculate FIT [23], which denotes the number of failures faced by the device operating for 10^9 *hours*. In our case, $FIT = soft\ error\ cross\ section \times reference\ neutron\ flux\ per\ hour \times 10^9$. Here, the reference neutron flux value of $12.946\ cm^{-2}h^{-1}$, obtained from the JESD89B standard [23], is used to calculate the FIT values for terrestrial applications, which is shown for the case study versions in the last row of Table 2. Various safety standards use FIT values to assess the safety levels of systems in the real world. For instance, the Automotive standard ISO 26262 [9, 52] requires the FIT values of the system to be no more than 10 to comply with their stringent Automotive Safety Integrity Level (ASIL) level D (ASIL-D) classification. From our case, if we assume all unique errors to result in catastrophic failures of interest, then the test setup running HDC10k inference fails to meet the ASIL-D safety level, but the test setup running HDC1024 inference complies with the ASIL-D safety level, with room to accommodate even more FIT rates associated with additional systems, if any.

4.3 Triple module redundancy and HDC1024 inference

Further improvement in reliability can be obtained by combining dimensionality reduction with industry-standard N-Module Redundancy (NMR) [28] techniques, while still consuming fewer resources than the higher dimension HDC inference algorithm. For instance, we analyzed the execution time and energy efficiency of HDC1024 with temporal Triple Module Redundancy (TMR) [16, 37, 55], where the HDC1024 inferences are executed with the same input thrice and the inference output is majority voted (statistical mode) as valid output. On average, this TMR version of HDC1024 consumed 1.1392 billion CPU cycles as execution time and 0.5627 joules as execution energy. Still, this TMR version of HDC1024 consumes 84.2% less execution time and 80.7% less execution energy than inferencing with HDC10k. TMR improves reliability, as the probability of the same error occurring in at least two of the three inference executions is usually much lower. The temporal TMR approach implemented for HDC1024 is comparable to the one proposed by Esposito and colleagues [16]. However, the voting mechanism was software implemented and inferences were executed successively.

Table 3. Overall results and parameters of HDC10k version and HDC1024 version, with their reduction due to reduced dimension

Measurements	HDC10k	HDC1024	Reduction
Total memory consumption ($bytes$)	146609	53771	63.3%
Average execution time per inference ($Giga\,CPU\,cycles$ @ $64\,MHz$)	7.20	0.40	94.4%
$Memory\,consumption \times execution\,time$ for VAT $\left(10^{14} \times bytes \times CPU\,cycles\right)$	10.60	0.22	98.0%
Soft error cross-section $\left(10^{-10} cm^2\right)$	9.18	2.63	71.4%
Model training accuracy (trade-off)	97.6%	94.8%	2.9%
Average energy consumption per inference ($Joules$ @ $3\,volts$)	2.92	0.18	93.8%

4.4 Vulnerable area-time, efficiency and reliability

The dimension of an HDC algorithm was reduced from 10k in the HDC10k version to 1024 in the HDC1024 version. This decreased a_r (memory consumption) by more than 63% and execution time t_e of HDC inference by 94.4%, causing an overall decrease in VAT by 98%.

Thus, the probability of neutrons impacting the HDC inference was shrunk, causing the soft error cross-section [23] to decrease by over 71%. While providing reliability and efficiency benefits, dimension-reduced HDC versions can maintain accuracies close to that of higher dimension HDC versions [39]. For instance, the difference in accuracy between the HDC10k version and the HDC1024 version was approximately 2.9%. This tradeoff is probably negligible in low-power Edge AI applications, as both the execution energy and execution time efficiency improved by about 94%. Thus, we practically proved our hypothesis from the real-world radiation experiments and efficiency analysis, whose overview is shown in Table 3. Figure 12 shows the key differences observed between the HDC1024 and the HDC10k, and the associated benefits of the dimension reduction.

Within the HDC community, higher dimensionality is closely associated with higher robustness, or reliability against single or multiple bit-flips as the proportion of allowable errors increases with HDC dimensionality [26]. Hence, the absolute amount of allowable errors (error threshold) will be lower for HDC1024 when compared to HDC10k, which

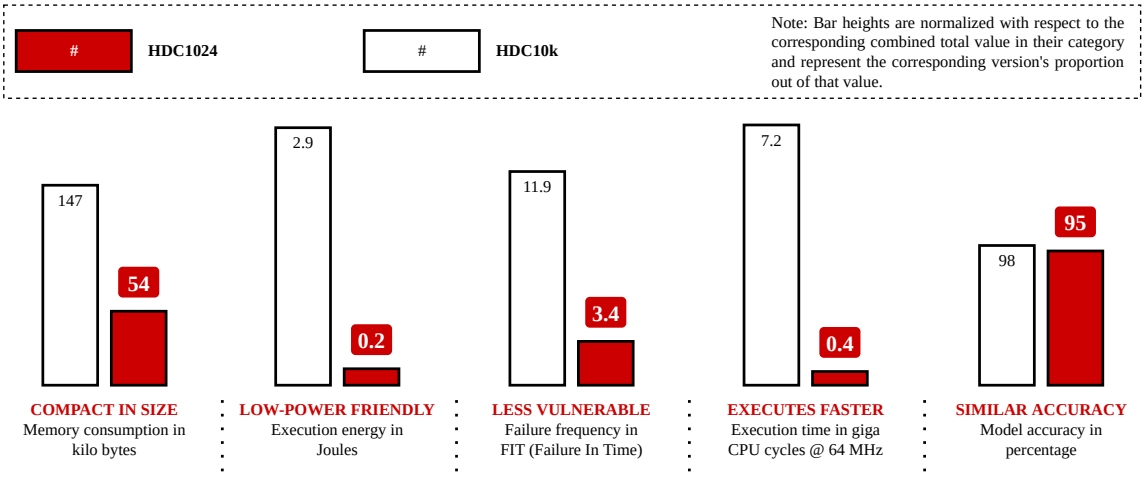


Fig. 12. Summary of parameters and their difference between AI algorithm versions HDC1024 and HDC10k.

is a tradeoff. Crossing such an error threshold can lead to mismatches in the predicted label, which must be avoided. However, we did not observe any predicted label mismatches between the experimental and golden inference outputs of HDC1024 and HDC10k, i.e., the errors were only observed in the prediction output and they were not able to affect the predicted label (cf., Section 1.2.2). This suggests that neither HDC1024 nor HDC10k surpassed the error threshold, enabling inferences to execute without predicted label mismatches during our radiation experiments. The dimension reduction for HDC1024 further reduces the number of bit-flips leading to errors, in turn reducing the probability of a scenario where HDC1024 crosses the error threshold under radiation.

Due to the holographic representation, when bits fail, the information degrades in relation to the number of failing bits irrespective of the position [26], and in some cases, reducing the factors such as execution time might not beneficially sway the absolute number of bit-flips affecting an HDC inference. Such cases can include readout or stuck-at errors in memories [38, 57], due to factors such as voltage over-scaling, and could benefit from higher HDC dimensionality. Even then, considering our case study versions, industry-standard techniques such as TMR could be a viable alternative, as they can be coupled with a lower dimension HDC1024 version to improve reliability while consuming less than 20% of the resources required for computing an inference with a higher dimensional HDC10k version.

Encoding schemes can also play a major role in improving the inference reliability against soft error-induced bit-flips. As also indicated in [46], encoding consumes more resources and can be optimized to reduce their execution time and runtime chip area; i.e., VAT. Should a bit-flip impact an HDC inference, the encoding scheme can still be beneficial in reducing that impact's effect on inference accuracy (to stay within the error threshold) by furthering the holographic representation of the HVs used and generated by it. This reduces the chance of a bit in a HV being more dominant than others, leaving no significant bits to flip.

4.5 General applicability

Others have studied dimension reduction [8, 58] using various datasets, including ISOLET, Fashion-MNIST, CTG, HAR, EEG, ERP, and the Parkinson's Disease digital biomarker DREAM challenge (PD Challenge), as well as the MNIST. They study various aspects of dimension reduction, for instance, the model accuracy and noise robustness. The study by Yan et al. [58] states that if the HV dimension d is sufficient to represent a vector with K classes (in particular, $d > log_2 K$), then the lower the dimension, the higher the accuracy. Another study by Basaklar et al. [8] presented a technique to minimize dimension while maintaining accuracy and improving robustness to noise in terms of the dissimilarity between class encoders. They also report other studies associated with dimension reduction. These studies could be used to potentially mitigate the tradeoffs reported in our study or even turn those tradeoffs into benefits.

The above studies associate lower dimensions with higher execution efficiency, which is in line with our observation. Due to beam time constraints explained in the third paragraph of Section 4.2, and the high costs associated with radiation experiments, we could not test several dimensions with various datasets. However, we tested two different dimensions of the HDC algorithm trained using the MNIST dataset, which showed that dimension reduction improved the reliability against radiation while considering the perturbations (i.e., mismatches) in the inference output. This perturbation is due to the soft error-induced bit-flip affecting the bits in HVs used by the HDC algorithm. When bits fail, the information degrades in relation to the number of failing bits, irrespective of their position [26] in an HV. This is applicable to HDC in general, irrespective of the dataset used for training. Such information loss likely decreases the reliability of the HDC inference output.

The chance of a bit failing within an HDC inference execution increases as more neutrons pass through the device within the execution time of an inference, measured in fluence. Within a given neutron flux, this fluence per inference

increases as the execution time increases. The complexity analysis of our HDC inference algorithm with respect to the dimension (N) revealed that both the memory in bits and the execution time required by a HDC inference increases linearly with N, i.e., our HDC inference algorithm has $O(N)$ complexity [6]. Hence, as the HDC dimension increases, the fluence per inference also increases, providing more neutrons to cause a bit-flip. Besides, the number of bits and the associated chip area usage [17] also increase with HDC dimension, which also contributes to the overall increase in the total number of neutrons impacting an inference, along with the increased execution time. This increases the chance of a bit failing within an inference due to neutrons, causing the reliability to decrease as the dimension increases. This is in line with our observation, and we expect this argument to hold even for a different set of dimensions within any given dataset. Experimentally verifying this argument for different dimensions and datasets, and finding the exact relationship between HDC dimension and reliability, requires additional radiation experiments, which are out of the scope of this work.

The test device chosen for the experiment also influences the observations. Hence, we chose a COTS processor, which has a general-purpose computing architecture, representative of the hardware used by a wide range of applications. For instance, the test device used in our experiments is based on the Cortex-M series of processors, which are typically used as microcontrollers for deeply embedded applications and targeted for intelligent edge devices [30]. The Cortex-M series is also the most popular family of low-power processors used in embedded wearable devices [8, 56]. These processors have several hours of lifetime with a small-scale battery, but they also have a resource-constrained architecture [56]. Hence, the effect of the test device in our experimental observations applies to a wide range of real-world low-power applications. Besides, experimenting with multiple test devices requires additional radiation experiments, which are out of the scope of this work (cf., Section 4.2).

5 CONCLUSION AND FUTURE WORK

As evident from the reliability and efficiency measurements supporting our hypothesis, a higher HDC dimensionality does not translate to higher reliability or robustness for mitigating the probability of a soft error-induced bit-flip affecting HDC inference. We validated our hypothesis by reducing HDC dimensionality from 10k to 1024 and conducting real-world experiments. For a negligible accuracy and error threshold tradeoff, which could be potentially mitigated further or even be turned into a benefit (cf., Section 4.5), we observed an 18 times improvement in execution time efficiency, a 16 times improvement in execution energy efficiency, and a 3.5 times improvement in inference reliability against soft error-induced bit-flips under neutron radiation (as soft error cross-section and FIT). These benefits are important in Edge AI applications where resources are limited and need to operate reliably.

The radiation experiment was performed with the neutron particle-based beam available at the ChipIr facility, making the results applicable at terrestrial and space levels since they are more realistic than several other fault injection-based methods [51]. The results are published [25] as they are costly to obtain and seldom available [36]. This can assist in validating the simulation parameters when using SFI-based HDC analysis.

Dimension reduction can be obtained by pruning a trained higher-dimension HDC model into a lower-dimension HDC model using approaches such as 'dimension-wise sparsity' [21] to avoid retraining the model from scratch. The dimension reduction can be coupled with standard redundancy techniques, such as TMR [55], for further improvements in the reliability of the HDC algorithms while consuming fewer resources than algorithms with higher dimensionality. Instruction Set Architecture (ISA) such as RISC-V [50] defines how the software and the hardware interact in a CPU or a family of CPUs and it's usually standardised. In future work, we plan to apply such standard ISA vector extensions for HDC computation and study the effect on HDC inference reliability.

Future radiation experiments with various HDC dimensions will improve the statistical significance of our results and aid in analyzing the relationship between HDC dimensionality and reliability against soft error-induced bit-flips. This relationship can aid in developing more accurate error models for easier reliability assessment of a given HDC model through simulation. Currently, VAT estimates can indicate whether the dimensional changes introduced in the HDC inference will increase or decrease the soft error cross-section, without requiring radiation experiments. However, future analysis of radiation experiment results could also provide more insights into the exact relationship between VAT, unique errors, memory consumption and soft error cross-section, and devise more effective methods to estimate VAT and a_r, which should also consider the compute area and the soft error sensitivity of various parts of the chip. This can be achieved by further analyzing the computing resources used for each phase of the HDC algorithm. Using a variety of datasets for HDC inference during radiation experiments could demonstrate the broad applicability of dimension reduction for reliability against soft error-induced bit-flips. Future radiation experiments could be improved to detect more error types during analysis for broadening our understanding of the impact of soft error-induced bit-flips on HDC inference. Those experiments can be conducted at various test facilities with different test devices to cover various test conditions.

Future encoding schemes can utilize bio-inspired near-sensor signal processing [32] or other techniques for optimal performance of the Encode subprocess to improve the HDC efficiency, accuracy and reliability. Such an efficient encoding scheme can still be combined with the dimension reduction for reliability against soft error-induced bit-flips and efficiency of HDC inference as the optimization also impacts the Classify subprocess.

ACKNOWLEDGMENTS

Beam experiments were provided by the ChipIR team thanks to Christopher Frost, Carlo Cazzaniga, and Maria Kastriotou (DOI: 10.5286/ISIS.E.RB2200004-1). This work has been supported by the MOVIQ (Mastering Onboard Vision Intelligence and Quality) project funded by Flanders Innovation & Entrepreneurship (VLAIO) and Flanders Space (VRI) and has received co-funding from the European Union NextGenerationEU; and the European Union's Horizon 2020 research and innovation programme under the grant agreement № 101008126.

AUTHOR CONTRIBUTIONS

Anuj Justus Rajappa: Conceptualization; methodology; software; validation; formal analysis (lead); investigation; data curation; writing – original draft; writing - review & editing; visualization. **Laura Smets**: Conceptualization; writing - review & editing. **Philippe Reiter**: Conceptualization; writing - review & editing; supervision; project administration; funding acquisition. **Paolo Rech**: Resources; funding acquisition; supervision; data curation. **Ynte Vanderhoydonc**: Formal analysis. **Ritesh Kumar Singh**: Writing - review & editing. **Siegfried Mercelis**: Supervision; writing - review & editing. **Jeroen Famaey**: Supervision; funding acquisition; writing - review & editing.

REFERENCES

[1] ARM . (accessed on 03 June 2024). Chapter 9. Data Watchpoint and Trace Unit. Available online: https://developer.arm.com/documentation/ddi0439/b/Data-Watchpoint-and-Trace-Unit?lang=en.

[2] Nordic semiconductors . (accessed on 03 June 2024). nRF52840 DK. Available online: https://www.nordicsemi.com/Products/Development-hardware/nrf52840-dk.

[3] SEGGER . (accessed on 03 June 2024). SEGGER compiler. Available online: https://wiki.segger.com/SEGGER_compiler.

[4] ADI Icron. (accessed on 22 Nov 2024). 4-port USB 2.0 Ethernet LAN Extender System. Available online: https://www.icron.com/products/icron-brand/usb-extenders/lan/usb-2-0-ranger-2304ge-lan/.

[5] Hussam Amrouch, Mohsen Imani, Xun Jiao, Yiannis Aloimonos, Cornelia Fermuller, Dehao Yuan, Dongning Ma, Hamza E. Barkam, Paul R. Genssler, and Peter Sutor. 2022. Brain-Inspired Hyperdimensional Computing for Ultra-Efficient Edge AI. In *2022 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*. 25–34. https://doi.org/10.1109/CODES-ISSS55005.2022.00017

[6] Milad Avazbeigi. 2009. *An Overview of Complexity Theory.* Physica-Verlag HD, Heidelberg, 19–36. https://doi.org/10.1007/978-3-7908-2151-2_2

[7] Hamza Errahmouni Barkam, SungHeon Eavn Jeon, Sanggeon Yun, Calvin Yeung, Zhuowen Zou, Xun Jiao, Narayan Srinivasa, and Mohsen Imani. 2023. Invited Paper: Hyperdimensional Computing for Resilient Edge Learning. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. 1–8. https://doi.org/10.1109/ICCAD57390.2023.10323671

[8] Toygun Basaklar, Yigit Tuncel, Shruti Yadav Narayana, Suat Gumussoy, and Umit Y. Ogras. 2021. Hypervector Design for Efficient Hyperdimensional Computing on Edge Devices. arXiv:2103.06709 [cs.LG] https://arxiv.org/abs/2103.06709

[9] Gabriele Boschi, Donato Luongo, Duccio Lazzarotti, Hanna Shaheen, Hayk Grigoryan, Gurgen Harutyunyan, Samvel Shoukourian, and Yervant Zorian. 2019. Memory FIT Rate Mitigation Technique for Automotive SoCs. In *2019 IEEE International Test Conference (ITC)*. 1–6. https://doi.org/10.1109/ITC44170.2019.9000158

[10] Carlo Cazzaniga, Marta Bagatin, Simone Gerardin, ALESSANDRA COSTANTINO, and CHRISTOPHER D. FROST. 2021. First Tests of a New Facility for Device-Level, Board-Level and System-Level Neutron Irradiation of Microelectronics. *IEEE Transactions on Emerging Topics in Computing* 9, 1 (2021), 104–108. https://doi.org/10.1109/TETC.2018.2879027

[11] Carlo Cazzaniga and Christopher D. Frost. 2018. Progress of the Scientific Commissioning of a fast neutron beamline for Chip Irradiation. *Journal of Physics: Conference Series* 1021 (may 2018), 012037. https://doi.org/10.1088/1742-6596/1021/1/012037

[12] Cheng-Yang Chang, Yu-Chuan Chuang, Chi-Tse Huang, and An-Yeu Wu. 2023. Recent Progress and Development of Hyperdimensional Computing (HDC) for Edge Intelligence. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 13, 1 (2023), 119–136. https://doi.org/10.1109/JETCAS.2023.3242767

[13] Davide Chiesa, Massimiliano Nastasi, Carlo Cazzaniga, Marica Rebai, Laura Arcidiacono, Ezio Previtali, Giuseppe Gorini, and Christopher D. Frost. 2018. Measurement of the neutron flux at spallation sources using multi-foil activation. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 902 (2018), 14–24. https://doi.org/10.1016/j.nima.2018.06.016

[14] Andrea Coronetti, Rubén García Alía, Jan Budroweit, Tomasz Rajkowski, Israel Da Costa Lopes, Kimmo Niskanen, Daniel Söderström, Carlo Cazzaniga, Rudy Ferraro, Salvatore Danzeca, Julien Mekki, Florent Manni, David Dangla, Cedric Virmontois, Nourdine Kerboub, Alexander Koelpin, Frédéric Saigné, Pierre Wang, Vincent Pouget, Antoine Touboul, Arto Javanainen, Heikki Kettunen, and Rosine Coq Germanicus. 2021. Radiation Hardness Assurance Through System-Level Testing: Risk Acceptance, Facility Requirements, Test Methodology, and Data Exploitation. *IEEE Transactions on Nuclear Science* 68, 5 (2021), 958–969. https://doi.org/10.1109/TNS.2021.3061197

[15] Fernando Fernandes dos Santos, Philippe Navaux, Luigi Carro, and Paolo Rech. 2019. Impact of Reduced Precision in the Reliability of Deep Neural Networks for Object Detection. In *2019 IEEE European Test Symposium (ETS)*. 1–6. https://doi.org/10.1109/ETS.2019.8791554

[16] Stefano Esposito, Sehriy Avramenko, and Massimo Violante. 2016. On the consolidation of mixed criticalities applications on multicore architectures. In *2016 17th Latin-American Test Symposium (LATS)*. 57–62. https://doi.org/10.1109/LATW.2016.7483340

[17] Tegze P Haraszti. 2002. *CMOS memory circuits.* Springer New York, NY. https://doi.org/10.1007/b117067

[18] Alejandro Hernández-Cano, Namiko Matsumoto, Eric Ping, and Mohsen Imani. 2021. OnlineHD: Robust, Efficient, and Single-Pass Online Learning Using Hyperdimensional System. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 56–61. https://doi.org/10.23919/DATE51398.2021.9474107

[19] Qiang Huang and Jin Jiang. 2019. An overview of radiation effects on electronic devices under severe accident conditions in NPPs, rad-hardened design techniques and simulation tools. *Progress in Nuclear Energy* 114 (2019), 105–120. https://doi.org/10.1016/j.pnucene.2019.02.008

[20] Thomas Huybrechts, Philippe Reiter, Siegfried Mercelis, Jeroen Famaey, Steven Latré, and Peter Hellinckx. 2021. Automated Testbench for Hybrid Machine Learning-Based Worst-Case Energy Consumption Analysis on Batteryless IoT Devices. *Energies* 14, 13 (2021). https://doi.org/10.3390/en14133914

[21] Mohsen Imani, Sahand Salamat, Behnam Khaleghi, Mohammad Samragh, Farinaz Koushanfar, and Tajana Rosing. 2019. SparseHD: Algorithm-Hardware Co-optimization for Efficient High-Dimensional Computing. In *2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. 190–198. https://doi.org/10.1109/FCCM.2019.00034

[22] Intel. (accessed on 03 June 2024). What Is Intel® Advanced Matrix Extensions (Intel® AMX)? Available online: https://www.intel.com/content/www/us/en/products/docs/accelerator-engines/what-is-intel-amx.html.

[23] JEDEC. (accessed on 03 June 2024). Measurement and Reporting of Alpha Particle and Terrestrial Cosmic Ray-Induced Soft Errors in Semiconductor Devices, Std. JESD89B, Sep. 2021. Available online: https://www.jedec.org/system/files/docs/JESD89B.pdf.

[24] Joulescope. (accessed on 03 June 2024). Joulescope™ JS110 User's Guide. Available online: https://download.joulescope.com/docs/JoulescopeUsersGuide/JoulescopeUsersGuide_v1_1.pdf.

[25] Anuj Justus, Paolo Rech, ISIS Neutron, and Muon Source. 2024. Effect of dimension reduction optimization on HDC inference with nRF52840_DK under neutron radiation. https://doi.org/10.5281/zenodo.11124961

[26] Pentti Kanerva. 2009. Hyperdimensional Computing: An Introduction to Computing in Distributed Representation with High-Dimensional Random Vectors. *Cognitive Computation* 1, 2 (01 Jun 2009), 139–159. https://doi.org/10.1007/s12559-009-9009-8

[27] Behnam Khaleghi, Jaeyoung Kang, Hanyang Xu, Justin Morris, and Tajana Rosing. 2022. GENERIC: highly efficient learning engine on edge using hyperdimensional computing. In *Proceedings of the 59th ACM/IEEE Design Automation Conference* (San Francisco, California) *(DAC '22)*. Association

for Computing Machinery, New York, NY, USA, 1117–1122. https://doi.org/10.1145/3489517.3530669

[28] Koren and Su. 1979. Reliability Analysis of N-Modular Redundancy Systems with Intermittent and Permanent Faults. *IEEE Trans. Comput.* C-28, 7 (1979), 514–520. https://doi.org/10.1109/TC.1979.1675397

[29] Elena Kulida and Valentin Lebedev. 2020. About the Use of Artificial Intelligence Methods in Aviation. In *2020 13th International Conference "Management of large-scale system development" (MLSD)*. 1–5. https://doi.org/10.1109/MLSD49919.2020.9247822

[30] Liangzhen Lai, Naveen Suda, and Vikas Chandra. 2018. CMSIS-NN: Efficient Neural Network Kernels for Arm Cortex-M CPUs. arXiv:1801.06601 [cs.NE] https://arxiv.org/abs/1801.06601

[31] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324. https://doi.org/10.1109/5.726791

[32] Jiean Li, Ming Xin, Zhong Ma, Yi Shi, and Lijia Pan. 2021. Nanomaterials and their applications on bio-inspired wearable electronics. *Nanotechnology* 32, 47 (sep 2021), 472002. https://doi.org/10.1088/1361-6528/abe6c7

[33] Dehua Liang, Hiromitsu Awano, Noriyuki Miura, and Jun Shiomi. 2023. DependableHD: A Hyperdimensional Learning Framework for Edge-Oriented Voltage-Scaled Circuits. In *Proceedings of the 28th Asia and South Pacific Design Automation Conference* (Tokyo, Japan) *(ASPDAC '23)*. Association for Computing Machinery, New York, NY, USA, 416–422. https://doi.org/10.1145/3566097.3567886

[34] Dehua Liang, Hiromitsu Awano, Noriyuki Miura, and Jun Shiomi. 2023. A Robust and Energy Efficient Hyperdimensional Computing System for Voltage-scaled Circuits. *ACM Trans. Embed. Comput. Syst.* (sep 2023). https://doi.org/10.1145/3620671 Just Accepted.

[35] Yufan Lu, Xiaojun Zhai, Sangeet Saha, Shoaib Ehsan, and Klaus McDonald-Maier. 2020. A self-scrubbing scheme for embedded systems in radiation environments. In *2020 IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS)*. 1–4. https://doi.org/10.1109/IOLTS50870.2020.9159718

[36] Lucas Matana Luza, Frederic Wrobel, Luis Entrena, and Luigi Dilillo. 2022. Impact of Atmospheric and Space Radiation on Sensitive Electronic Devices. In *2022 IEEE European Test Symposium (ETS)*. 1–10. https://doi.org/10.1109/ETS54262.2022.9810454

[37] R. E. Lyons and W. Vanderkulk. 1962. The Use of Triple-Modular Redundancy to Improve Computer Reliability. *IBM Journal of Research and Development* 6, 2 (1962), 200–209. https://doi.org/10.1147/rd.62.0200

[38] Dongning Ma, Sizhe Zhang, and Xun Jiao. 2023. Robust Hyperdimensional Computing against Cyber Attacks and Hardware Errors: A Survey. In *Proceedings of the 28th Asia and South Pacific Design Automation Conference* (Tokyo, Japan) *(ASPDAC '23)*. Association for Computing Machinery, New York, NY, USA, 598–605. https://doi.org/10.1145/3566097.3568355

[39] Alec Xavier Manabat, Celine Rose Marcelo, Alfonso Louis Quinquito, and Anastacia Alvarez. 2019. Performance Analysis of Hyperdimensional Computing for Character Recognition. In *2019 International Symposium on Multimedia and Communication Technology (ISMAC)*. 1–5. https://doi.org/10.1109/ISMAC.2019.8836136

[40] Massimo Merenda, Carlo Porcaro, and Demetrio Iero. 2020. Edge Machine Learning for AI-Enabled IoT Devices: A Review. *Sensors* 20, 9 (2020). https://doi.org/10.3390/s20092533

[41] Multicomp pro. (accessed on 03 June 2024). Bench Top Remote Programmable Linear Mode Power Supply (MP710078). Available online: https://www.farnell.com/datasheets/2830052.pdf.

[42] Patrick Nsengiyumva. 2014. INVESTIGATING THE EFFECTS OF SINGLE-EVENT UPSETS IN STATIC AND DYNAMIC REGISTERS. https://scholars.unh.edu/thesis/984

[43] Patrick Nsengiyumva and Qiaoyan Yu. 2015. Investigation of single-event upsets in dynamic logic based flip-flops. In *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*. 818–821. https://doi.org/10.1109/ISCAS.2015.7168759

[44] Paul A. Oche, Gideon A. Ewa, and Nwanneka Ibekwe. 2024. Applications and Challenges of Artificial Intelligence in Space Missions. *IEEE Access* 12 (2024), 44481–44509. https://doi.org/10.1109/ACCESS.2021.3132500

[45] Prathyush Poduval, Zhuowen Zou, Hassan Najafi, Houman Homayoun, and Mohsen Imani. 2021. StocHD: Stochastic Hyperdimensional System for Efficient and Robust Learning from Raw Data. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*. 1195–1200. https://doi.org/10.1109/DAC18074.2021.9586166

[46] Abbas Rahimi, Pentti Kanerva, and Jan M. Rabaey. 2016. A Robust and Energy-Efficient Classifier Using Brain-Inspired Hyperdimensional Computing. In *Proceedings of the 2016 International Symposium on Low Power Electronics and Design* (San Francisco Airport, CA, USA) *(ISLPED '16)*. Association for Computing Machinery, New York, NY, USA, 64–69. https://doi.org/10.1145/2934583.2934624

[47] Anuj Justus Rajappa, Philippe Reiter, Tarso Kraemer Sarzi Sartori, Luiz Henrique Laurini, Hassen Fourati, Siegfried Mercelis, Jeroen Famaey, and Rodrigo Possamai Bastos. 2023. SMART: Selective MAC zero-optimization for neural network reliability under radiation. *Microelectronics Reliability* 150 (2023), 115092. https://doi.org/10.1016/j.microrel.2023.115092 Special issue of 34th European Symposium on Reliability of Electron Devices, Failure Physics and Analysis, ESREF 2023.

[48] A. Ramos, A. Carrasco, J. Fontanet, L.E. Herranz, D. Ramos, M. Díaz, J.M. Zazo, O. Cabellos, and J. Moraleda. 2024. Artificial intelligence and machine learning applications in the Spanish nuclear field. *Nuclear Engineering and Design* 417 (2024), 112842. https://doi.org/10.1016/j.nucengdes.2023.112842

[49] RISC-V. (accessed on 03 June 2024). Andes Announces RISC-V Multicore 1024-Bit Vector Processor: AX45MPV. Available online: https://riscv.org/news/2022/12/andes-announces-risc-v-multicore-1024-bit-vector-processor-ax45mpv/.

[50] RISC-V. (accessed on 03 June 2024). Specifications. Available online: https://riscv.org/technical/specifications/.

[51] Annachiara Ruospo, Ernesto Sanchez, Lucas Matana Luza, Luigi Dilillo, Marcello Traiola, and Alberto Bosio. 2023. A Survey on Deep Learning Resilience Assessment Methodologies. *Computer* 56, 2 (2023), 57–66. https://doi.org/10.1109/MC.2022.3217841

[52] Jose Serna, Simon Diemert, Laure Millet, Rami Debouk, Ramesh S, and Jeffrey Joyce. 2020. Bridging the Gap between ISO 26262 and Machine Learning: A Survey of Techniques for Developing Confidence in Machine Learning Systems. *SAE International Journal of Advances and Current Practices in Mobility* 2, 3 (apr 2020), 1538–1550. https://doi.org/10.4271/2020-01-0738

[53] Laura Smets, Werner Van Leekwijck, Ing Jyh Tsang, and Steven Latré. 2023. An Encoding Framework for Binarized Images using HyperDimensional Computing. arXiv:2312.00454 [cs.CV]

[54] Laura Smets, Werner Van Leekwijck, Ing Jyh Tsang, and Steven Latré. 2023. Training a Hyperdimensional Computing Classifier Using a Threshold on Its Confidence. *Neural Computation* 35, 12 (11 2023), 2006–2023. https://doi.org/10.1162/neco_a_01618 arXiv:https://arxiv.org/abs/2305.19007

[55] J.F. Wakerly. 1976. Microcomputer reliability improvement using triple-modular redundancy. *Proc. IEEE* 64, 6 (1976), 889–895. https://doi.org/10.1109/PROC.1976.10239

[56] Xiaying Wang, Michael Hersche, Batuhan Tömekce, Burak Kaya, Michele Magno, and Luca Benini. 2020. An Accurate EEGNet-based Motor-Imagery Brain–Computer Interface for Low-Power Edge Computing. In *2020 IEEE International Symposium on Medical Measurements and Applications (MeMeA)*. 1–6. https://doi.org/10.1109/MeMeA49120.2020.9137134

[57] Weihong Xu, Viji Swaminathan, Sumukh Pinge, Sean Fuhrman, and Tajana Rosing. 2023. HyperMetric: Robust Hyperdimensional Computing on Error-prone Memories using Metric Learning. In *2023 IEEE 41st International Conference on Computer Design (ICCD)*. 243–246. https://doi.org/10.1109/ICCD58817.2023.00045

[58] Zhanglu Yan, Shida Wang, Kaiwen Tang, and Weng-Fai Wong. 2023. Efficient Hyperdimensional Computing. In *Machine Learning and Knowledge Discovery in Databases: Research Track*, Danai Koutra, Claudia Plant, Manuel Gomez Rodriguez, Elena Baralis, and Francesco Bonchi (Eds.). Springer Nature Switzerland, Cham, 141–155. https://doi.org/10.1007/978-3-031-43415-0_9

[59] Sizhe Zhang, Mohsen Imani, and Xun Jiao. 2022. ScaleHD: Robust Brain-Inspired Hyperdimensional Computing via Adapative Scaling. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design* (San Diego, California) *(ICCAD '22)*. Association for Computing Machinery, New York, NY, USA, Article 31, 9 pages. https://doi.org/10.1145/3508352.3549376

[60] Sizhe Zhang, Ruixuan Wang, Jeff Jun Zhang, Abbas Rahimi, and Xun Jiao. 2021. Assessing Robustness of Hyperdimensional Computing Against Errors in Associative Memory : (Invited Paper). In *2021 IEEE 32nd International Conference on Application-specific Systems, Architectures and Processors (ASAP)*. 211–217. https://doi.org/10.1109/ASAP52443.2021.00039

Received 03 June 2024; first revision 22 November 2024; second revision 07 June 2025