Assessing Robustness of Hyperdimensional Computing Against Errors in Associative Memory

(Invited Paper)

Sizhe Zhang*, Ruixuan Wang*, Jeff (Jun) Zhang‡, Abbas Rahimi§, Xun Jiao*
*Villanova University, ‡Harvard University, §IBM Research – Zurich

Abstract—**Brain-inspired hyperdimensional computing (HDC) is an emerging computational paradigm that has achieved success in various domains. HDC mimics brain cognition and leverages hyperdimensional vectors with fully distributed holographic representation and (pseudo)randomness. Compared to the traditional machine learning methods, HDC offers several critical advantages, including smaller model size, less computation cost, and one-shot learning capability, making it a promising candidate in low-power platforms. Despite the growing popularity of HDC, the robustness of HDC models has not been systematically explored. This paper presents a study on the robustness of HDC to errors in associative memory—the key component storing the class representations in HDC. We perform extensive error injection experiments to the associative memory in a number of HDC models (and datasets), sweeping the error rates and varying HDC configurations (i.e., dimension and data width). Empirically, we observe that HDC is considerably robust to errors in the associative memory, opening up opportunities for further optimizations. Further, results show that HDC robustness varies significantly with different HDC configurations such as data width. Moreover, we explore a low-cost error masking mechanism in the associative memory to enhance its robustness.**

I. Introduction

Compared to traditional machine learning methods, an emerging memory-centric computational paradigm called hyperdimensional computing (HDC) [16] has shown advantages such as smaller model size, less computation cost, and one-shot learning capability. HDC mimics the working mechanism of the brain by computing wide and abstract patterns of neural activities instead of scalar numbers. Recently, HDC has demonstrated promising capability in a wide range of classification tasks dealing with raw characters [30], or extracted features from voice [9], time-series [29], and spikes [26].

Moreover, HDC gracefully degrades under low signal-to-noise ratio conditions (see [27] for an overview). For instance, compared to the baseline language classifier, HDC tolerates $9\times$ higher intermittent errors in the digital memory cells [30], or $\approx 83\times$ higher hard errors in the resistive RAMs [21] and carbon nanotube logic [39]; it can also recover from these permanent faults by unsupervised regeneration of new blocks for the associative memory to compensate the accuracy loss [7]. In a large-scale demonstration, HDC achieves comparable accuracies to software implementations for three classification tasks using 760,000 phase-change memory devices exhibiting significant spatial variability and noise [17]. In a similar vein, HDC maintains the top accuracy in various few-shot learning problems while withstanding up to 30% conductance variations in phase-change memory devices [18]. Nevertheless, the reliability and robustness of HDC models have not been systematically explored yet. This paper aims to systematically explore and quantify the robustness of HDC classifiers to errors in associative memory, opening up new directions for further optimizations.

Conventional computing typically requires a near-perfect execution with negligible error rates (e.g., $< 10^{-15}$ [10]) to guarantee correctness. Such requirement, however, poses a high design cost at device and architecture levels. Recently, designers are devising a scenario to continuing the operations even in the presence of errors, leading to *approximate computing* which allows occasional errors in a system to appear as long as it delivers acceptable output quality [11], [25]. Machine learning methods, especially deep neural networks (DNNs), are known to be more robust than conventional workloads [12], [22], [41]. Leveraging such error resilience, designers have already improved the efficiency of DNN systems on various platforms such as ASICs [41] and FPGAs [34].

Such exploration in DNNs also motivates us to systematically understand and explore the extent of error resilience of HDC models, potentially allowing designers to push the design guardband for digital HDC systems safely. This includes increased clock frequency to boost throughput and/or reduced supply voltage to improve energy efficiency at the architecture/circuit level [41], [19]. In this paper, we perform a timely investigation on the error robustness of HDC digital systems, and seek to address the following research questions (**RQs**):

RQ1: How robust is HDC to errors in associative memory in digital hardware (e.g., timing errors due to voltage scaling or soft errors due to high-energy particles)?

RQ2: Does the impact of those hardware errors on HDC classification accuracy vary across different HDC configurations and datasets?

RQ3: Is there an efficient way to mitigate the impact of hardware errors on the associative memory and to improve the robustness of the HDC system?

To this end, we systematically *measure*, *quantify*, and *characterize* the impact of errors in associative memory on the classification accuracy of HDC models. Such characterization of error resilience can be broadly applied across various devices and architectures that use HDC. Specifically, we make the following major contributions:

- We broadly investigate various HDC classifiers across application domains including voice classification, human activity recognition, and medical diagnosis. We perform extensive error injection experiments to mimic common sources of hardware errors, including soft errors and timing errors.
- We measure, characterize, and quantify the impact of errors on the classification accuracy of HDC models. We evaluate the robustness of HDC against errors in associative memory across datasets, configurations, and error rates. Our results show that HDC is considerably robust to errors in associate memory across model configurations and datasets.
- We explore a low-cost error mitigation mechanism for the associative memory in HDC by detecting and masking the corrupted bits to zero, which can effectively improve the robustness of HDC by up to 13X across different model configurations and accuracy loss constraints.

II. Related Works

Hyperdimensional Computing Existing works on HDC focus mainly on the application of HDC and optimization of HDC processing. In particular, HDC has been used in modern robotics to perform active perception by integrating the sensory perceptions experienced by an agent with its motoric capabilities, which is vital to autonomous learning agents [26]. HDC exhibits 97.8% accuracy on hand gesture recognition, which surpasses support vector machine by 8.1% [28]. HDC has also achieved success in the language classification [30], [36], drug discovery [24], anomaly detection [38], and voice recognition [9]. In specific HDC encoding, instead of binary output, multi-bit quantization is applied to the output vectors [8]. In-memory computing for HDC systems has been proposed based on the small-scale resistive RAMs [21] and large-scale phase-change memory devices [17]. There are also optimizations on HDC targeted at different architectures such as FPGAs [35] and monolithic 3D IC [39].

Error Resilience of DNNs Safety-critical systems, e.g., mobile robots, raise strong reliability requirements on machine learning techniques prior to any practical hardware/software deployment. Thus, some recent research efforts have focused on the robustness of DNNs [12], [20], [32], [41], [19]. These studies focus on errors arising across the hardware/software computing stack and the consequent impacts on the classification/object detection accuracy at the application level. Jiao et al. presented the first study of DNN vulnerability to timing errors caused by fluctuations in operating conditions. Li et al. empirically evaluated DNNs at the software level, and found that the error resilience of a DNN system depends on data types, values, data reuses, and types of layers in the design [20]. In the same spirit, *Ares* [32] proposed a lightweight, scalable fault injection framework that can be accelerated on GPUs to explore larger design space. *ThunderVolt* [41] performed aggressive supply voltage underscaling on a systolic array at the gate level and evaluated the impact of circuit timing error on DNNs accuracy. *MATIC* [19] studied the impact of memory-related errors on deep learning accelerators.

Our Work While there are papers examining the algorithm-level robustness of HDC to adversarial inputs [23], [37], [40], there lacks a systematic investigation on HDC robustness to hardware errors. This paper presents a timely effort in such a paradigm by extensively injecting errors to associative memory of HDC models. Our aim is to offer timely insights and early guidance for designing efficient and reliable HDC systems.

III. HDC Model Development

In this section, we describe the HDC model development process including the **Encoding**, **Training**, **Retraining**, and **Inference** stages. Without loss of generality, we use the example of voice recognition to introduce the process of constructing HDC models. Note that the proposed HDC architecture illustrated in Fig. 1 can extend to other applications.

A. Hypervector (HV) and HDC arithmetic

Hypervector (HV) is a type of high-dimensional (usually with a $D = 10000$), holographic, and (pseudo-)random vector with independent and identically distributed (i.i.d.) components, where each component is a binary $(0/1)$ or a bipolar $(-1/+1)$ number. In HDC, HVs are the fundamental blocks to represent "items" reflecting real-life matters in classification tasks, such as pixel values, signal levels, or language characters. When the dimensionality is sufficiently high (e.g., $D = 10,000$), any two random HVs are nearly orthogonal [16]. HDC utilizes different operations HVs support as means of producing aggregations of information or creating representations of new information. In general, HV supports three types of arithmetic operations: (element-wise) addition, (element-wise) multiplication and permutation (cyclic shifting). Additions and multiplications take two input HVs as operands and perform **element-wise** add or multiply operations on the two HVs. Permutation takes one HV as the input operand and perform **cyclic rotation**. All the operations do not modify the dimensionality of the input HVs, i.e. the input and the output HVs are in the same dimension. Multiplication and permutation will produce HVs that are orthogonal to the original operand HVs while addition will preserve 50% information of each original operand HVs [16].

B. Stage 1: Encoding

The encoding stage is the first stage of HDC. Different applications may use different encoding methods. In this paper, we use the record-based encoding [5] across all three datasets. The detail of the encoding process is shown in Fig. 1 encoding part. First, we normalize all the input feature values to the range of $[-1, 1]$ and quantize the entire range to k levels, so each feature value now corresponds to a specific level. We will generate k random bipolar HVs (referred to as "level HVs") so that each level corresponds to a specific level HV. For example, if we quantize the $[-1, 1]$ to 100 levels, we will generate 100 level HVs. Second, depending on the number of feature values (e.g., n) in each input sample, we will generate n random

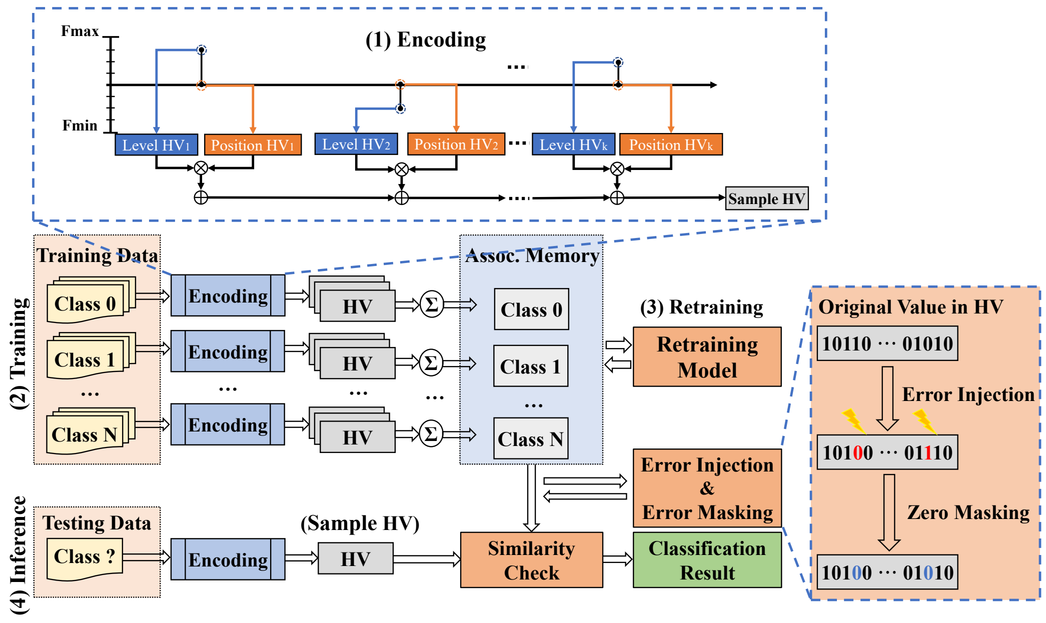


Fig. 1. An example HDC classifier on alphabet voice recognition dataset with four stages: (1) **Encoding**, (2) **Training**, (3) **Retraining** and (4) **Inference**. **Encoding** stage maps or encodes a voice signal into a representative HV (sample HV). **Training** stage sums all the sample HVs of the same class (e.g., all signal HVs corresponding to "a") to generate class HVs in the associative memory. **Retraining** stage will update associate memory to enhance the accuracy of the classifier. **Inference** stage performs classification for a given testing voice signal by performing similarity check between sample HV and class HVs in the associative memory. Note that errors are injected at the associative memory during the inference stage.

bipolar HVs (referred to as "position HVs") so that each position corresponds to a specific position HV. For example, there are 617 features in each ISOLET data sample, so we generate 617 position HVs. Third, for each feature value in an input sample, we multiply its level HV (corresponding to the feature value) and its position HV (corresponds to the feature position) to create a new HV representing the information of this specific feature. For each ISOLET input sample, we would create 617 HVs by binding the information of the feature value and feature position. Then, we add all the 617 newly-created HVs together to create another HV (referred to as "sample HV"), which represents the information of the whole sample.

$$\begin{aligned} \vec{H_f} &= \vec{H_{position}} * \vec{H_{level}} \\ \vec{H_{sample}} &= \vec{H_{f_1}} + \vec{H_{f_2}} \dots \vec{H_{f_n}} \end{aligned} \tag{1}$$

C. Stage 2: Training

The encoding stage encodes each input sample into a sample HV. The training stage simply add all the sample HVs of the same class together to create a class HV representing that specific class. For example, in ISOLET dataset, it will add all the sample HVs of letter "a" together to create a class HV representing letter "a". The same process applies to other letters by aggregating the information from their sample HVs. In the end, we would generate 26 class HVs and store them in an associative memory which will be used for inference.

D. Stage 3: Retraining

Before using the trained associative memory for inference, we can perform another optional step called retraining to enhance the classification accuracy of the HDC classifier. The process is as follows: we perform classification on each training data sample, and if the classification for a certain sample is wrong, the associative memory will update its value by adding the sample HV $\vec{H_t}$ into the ground-truth class HV $\vec{A^r}$ and subtract the sample HV $\vec{H_t}$ from the wrongly-predicted class HV $\vec{A^w}$, as shown in Eq. 2. Retraining can be done for multiple epochs over the entire training dataset.

$$\begin{aligned} \vec{A^w} &= \vec{A^w} - \vec{H_t} \\ \vec{A^r} &= \vec{A^r} + \vec{H_t} \end{aligned} \tag{2}$$

E. Stage 4: Inference

The inference stage uses the trained associative memory to classify unseen signals from the testing dataset. First, same as the training stage, the encoding stage encodes the testing sample into a sample HV (referred to as "query HV"). Second, the HDC classifier checks the similarity between the query HV and every class HV in the associative memory. In this paper, similarity is measured using cosine similarity between two HVs (HV_1 and HV_2): $Sim(HV_1, HV_2) = 1 - \frac{HV_1 \cdot HV_1}{||HV_1|| ||HV_2||}$, where HV_1 is the query HV and HV_2 is the class HV. Finally, the class with the highest similarity with the query HV will be predicted as the class of the signal.

F. Quantization

The associative memory contains the final class HVs represent all the classes. Note that according to Section III-C, the class HV is the sum of all the sample HVs that belong to the same class. Therefore, typically speaking, each element in the class HV is an integer that needs to be represented and stored using multiple bits. In this paper, to investigate the impact of data-width on HDC robustness, we employ the quantization process to use less bits to store each element. For example, we represent each element using 16 bits, 8 bits, and 1 bit (binarize), by quantizing each element to a certain level within its range. For example, we can binarize each element in the HV using Equation 3. Note that this process may incur accuracy loss as presented later.

$$HV[i] = \begin{cases} 1, & HV[i] > HV_{mean} \\ 0, & HV[i] \leq HV_{mean} \end{cases} \tag{3}$$

IV. Error Injection

In this section, we describe the types of errors we aim to inject to HDC models and present the error models we will use to mimic these error types.

A. Sources of Hardware Errors

With technology scaling, modern computing system suffer from faults occurred at various hardware levels [31]. In this paper, we consider two most common errors: timing errors [4], [15], [13], [14] and soft errors [6], [20]. Timing errors can often be caused by microelectronic variations from several sources: (1) manufacturing variation induced by random dopant fluctuations and variations in channel length (L) and threshold voltage (V_{th}); (2) aging and wear-out effects such as negative bias temperature instability (NBTI), electromigration, and time-dependent dielectric breakdown; and (3) variations in operating conditions, which is typically caused by supply voltage droops and temperature fluctuations. Note that designers can also aggressively underscale the supply voltage to save chip's energy [41], [33]. Due to the burgeoning use of edge intelligence in mobile and autonomous systems, the threat from variations in operating conditions is continually increasing.

Soft errors are often a manifestation of transient faults, which are caused by alpha particles, cosmic rays, thermal neutrons, or other environmental causes. While being temporary, soft errors can corrupt a memory location or a register and produce incorrect results until it is updated. This is typically referred as single-event upset (SEU), which appears randomly during the program execution. The continuing reduction in transistor sizes with technology scaling is increasing the susceptibility of modern systems to soft errors. For example, the soft error rate per bit is estimated to increase 8% with each technology scaling by semiconductor vendors [6].

B. Error Model

Due to HDC's memory-centric approach, we focus on errors that occurred in associative memory in this paper. Existing studies on DNN resilience follow the same practice since storage elements have shown much higher sensitivity to soft errors than logic elements [32], [20]. We use the most widely-used error models, i.e., single bit flip (**SBF**) model [20], [32] to mimic the hardware errors. The SBF model is most widely used to model timing/soft errors: it randomly flips a single bit upon the occurrence of an error. We inject errors to HDC's associate memory with the pre-defined error rates and error models during the inference stage.

TABLE I
Error-free accuracy for different datasets and configuration.

Configuration	16-bit			8-bit			1-bit		
	10000	5000	3000	10000	5000	3000	10000	5000	3000
ISOLET	0.9442	0.9448	0.9397	0.9442	0.9455	0.9403	0.9134	0.8864	0.8512
HAR	0.9377	0.9358	0.9393	0.9171	0.9275	0.9326	0.8716	0.8646	0.8520
CARDIO	0.9343	0.9296	0.9155	0.9296	0.9296	0.8638	0.8685	0.8216	0.7793

V. Experimental Results

A. Experimental Setup

We use three different HDC configurations in dimensions: 10000, 5000, and 3000, and three different data width: 16-bit, 8-bit, and 1-bit. The different HDC configurations and their corresponding (error-free) accuracy are presented in Table I. Note that using $D = 5000$ and 8-bit data width incur negligible accuracy loss but further dropping to $D = 3000$ and 1-bit data width could present a visible accuracy loss. In any case, we focus on the relative accuracy drop after injecting errors. For each configuration, we sweep 9 error rates equally spaced on a logarithmic scale from 10^{-9} to 10^{-1}, e.g., 10^{-9}, 10^{-8}, etc. At each error rate, we repeat our experiment 10 times and report the mean accuracy. For each model, we apply different retraining learning rates and epochs to get a fine-tuned model.

We use three datasets as follows:

- Speech Recognition (**ISOLET** [3]) to recognize voice audio of the 26 letters of the English alphabet. This dataset contains 150 subjects speaking the name of each alphabet letter twice. We use 6,238 samples for training and 1,559 samples for inference;
- Human Activity Recognition (**HAR** [1]) to recognise 12 kinds of human activities. The dataset is built from 30 subjects performing activities of daily living (ADL) with 561 features. We used 7,767 signal data for training and 3162 for inference;
- Medical Diagnosis (**CARDIO** [3]) to classify measurements of fetal heart rate (FHR) and uterine contraction (UC) features into 10 classes (label consensus by obstetricians). It has a total of 2,126 fetal cardiotocograms (CTGs) signals.

B. Robustness of HDC to Errors in Associative Memory

Robustness across application datasets: As shown in Fig. 2 (a) to Fig. 2 (i), all HDC classifiers for different applications present a certain degree of robustness to errors in the associative memory. Across all different configurations and applications, HDC classifiers can tolerate at least 10^{-6} error rate, which is significantly more robust than conventional

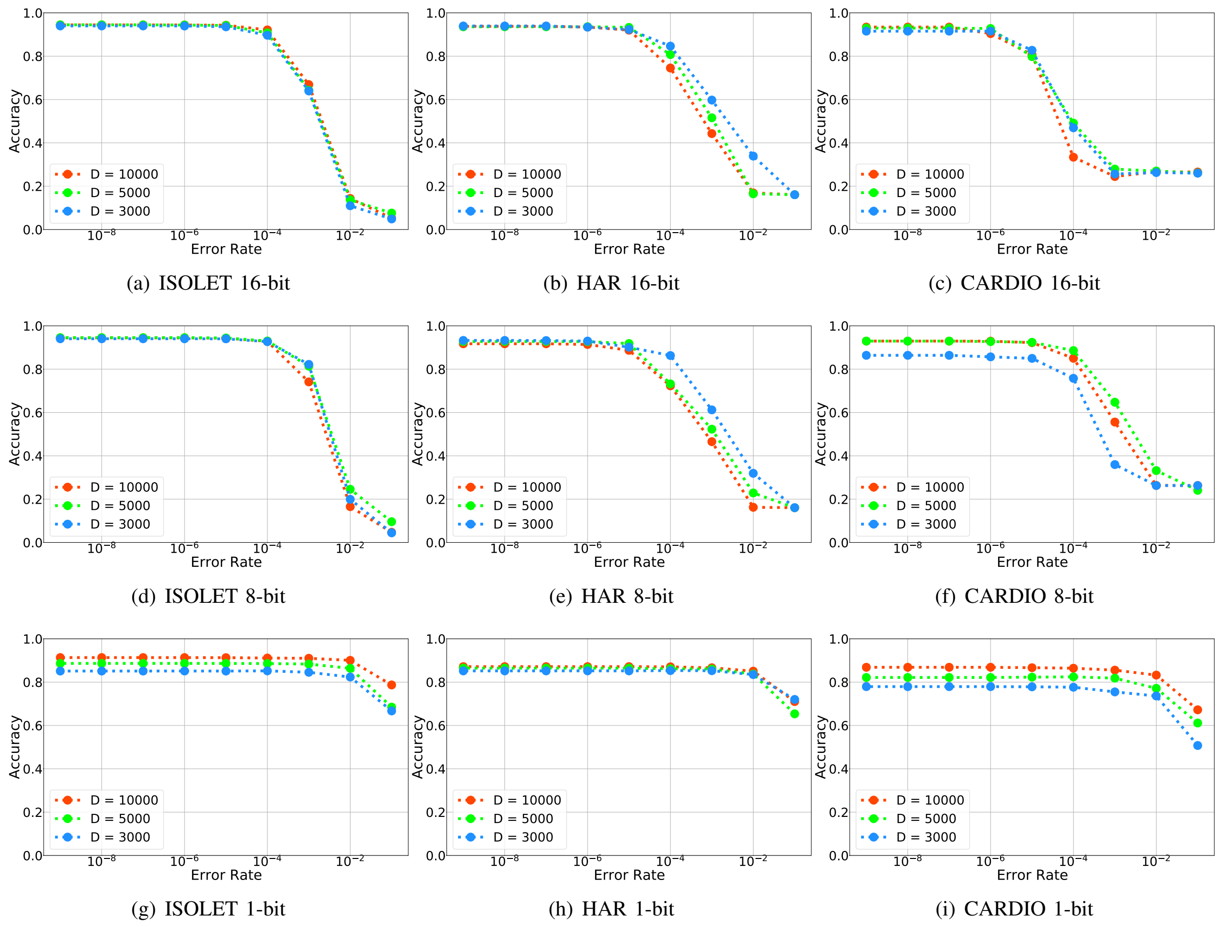


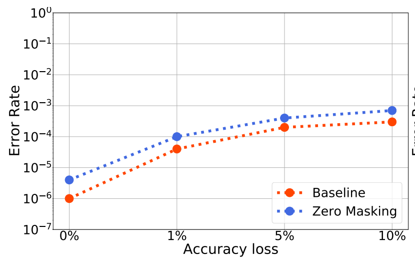
Fig. 2. Error resilience of HDC models for three datasets across different dimensions with different associate memory data width.

computing ($< 10^{-15}$) and is on a par with neural networks (10^{-9} to 10^{-6}) [32]. Further, we can also observe different applications present different robustness. For example, as shown in Fig. 2 (d), (e), (f), the accuracy for ISOLET does not drop even with an error rate of 10^{-4}, while for HAR and CARDIO, the accuracy starts to see notable drop at 10^{-5} error rate.

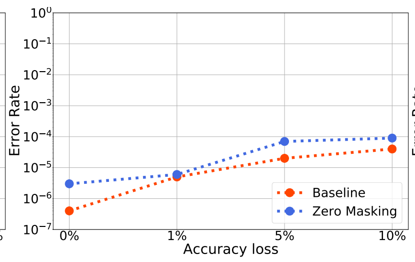
The differences between applications are reasonable because HDC model is designed highly tied to the application in multiple aspects such as the task difficulty and the encoding scheme. A detailed error characterization at the application level can enable HDC system designers to better understand the inherent error resilience for a certain application, guide the design to optimize the operating regimes according to the error characteristics, and explore the trade-offs such as energy efficiency vs. accuracy.

Robustness across dimensions: The dimension configuration will also affect the robustness of HDC classifiers. As shown in **HAR** and **ISOLET**, we can observe that lower dimensions can provide more robustness for HDC classifiers. One explanation to this is that, under the same error rate, lower dimensions implies fewer errors in the HVs. This observation is also critical to HDC classifier design in the sense that just-enough dimensions should be used to provide higher robustness. Further, lower dimension can also reduce the hardware costs and runtime overhead.

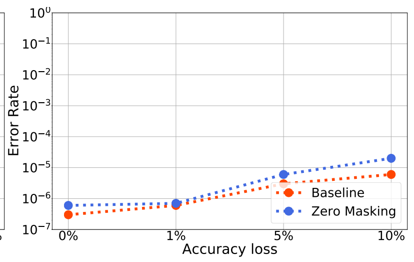
Robustness across data-width: It is observed that generally less data-width leads to an improved robustness. By conducting a row-to-row comparison in Fig. 2, we can observe that 8-bit HDC is more robust than 16-bit HDC, and 1-bit HDC, although having lowest error-free baseline accuracy, is the most robust configuration. For example, in 1-bit HDC, the accuracy does not see a notable drop until 10^{-2} error rate. The reason to such phenomenon is that, the more bits we use to represent a number, the more significant are the high-order bits. Thus, if there is a bit flip occurring in such high-order bits, it will create significant impact. For example, for a 16-bit HDC, if the most significant bit (MSB) has a bit flip, the magnitude change is up to 2^{15}, while for a 8-bit HDC, the same bit flip would only lead to a magnitude change of 2^7. This observation is critical in HDC design when considering the robustness to errors in the sense that HDC should use "just enough" data width (or precision) to accommodate application-specific data. This is more hardware friendly as well as more error robust.



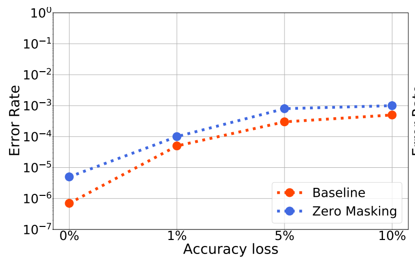
(a) ISOLET 16-bit



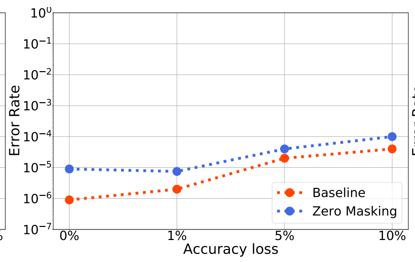
(b) HAR 16-bit



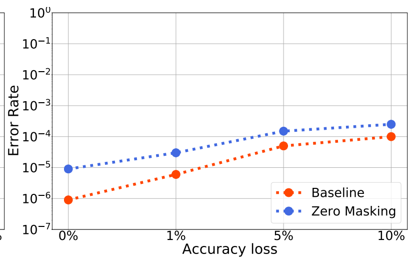
(c) CARDIO 16-bit



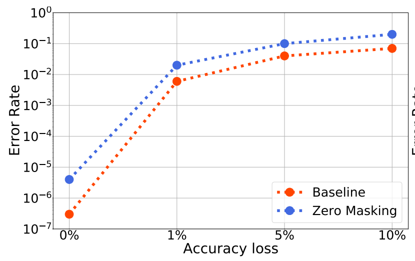
(d) ISOLET 8-bit



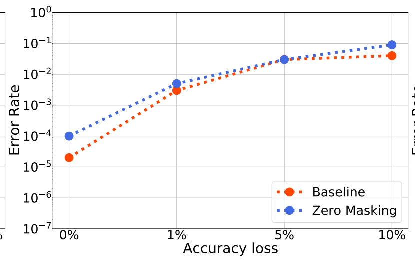
(e) HAR 8-bit



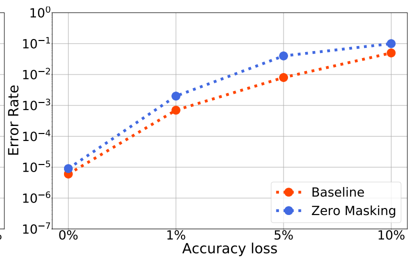
(f) CARDIO 8-bit



(g) ISOLET 1-bit



(h) HAR 1-bit



(i) CARDIO 1-bit

Fig. 3. Error resilience of HDC models with/without zero masking across different data width (D = 10000). Four data points in each figure is max error rate that can be tolerated within 0%, 1%, 5% and 10% accuracy loss.

C. *Low-cost Error Masking*

To mitigate the impact of hardware errors on HDC model performance, we explore low-cost error masking techniques to detect and mask errors. A simple error detection circuitry based on double value sampling can be applied, such as Razor [2], which has been used for timing error monitoring. Upon the detection of an error, we will mask the error by setting to corresponding bit(s) to 0, which is also illustrated in Fig. 1. The detailed circuit-level implementations of the detection circuitry can be found in [2], [33], which is out of scope for this study. Fig. 3 shows the results when errors are detected and masked by our error masking scheme. By using this error masking, it strengthens the robustness of the HDC up to 13X across all our experiment. This shows that combing this error masking method with HDC can significantly gain for error resilience.

VI. Conclusion

Brain-inspired HDC is an emerging computational paradigm but is gaining increasing capability in various problem domains. This paper presents a timely effort in systematically exploring and characterize the error robustness of HDC by performing extensive error injection experiments. We vary a wide range of error rates, datasets, and HDC configurations (i.e., dimensions). Experimental results show that HDC is considerably robust to hardware errors, which can be leveraged to optimize the accuracy-energy-performance design trade-off. Dimension and data width can also influence the robustness of HDC model. We further investigate a low-cost error mitigation mechanism that can improve the error resilience of HDC models by up to 13X. While HDC performance still expects more advancements both in theoretical and implementation aspects, this paper aims to shed light on robustness and reliability in HDC developments.

References

[1] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *Esann*, volume 3, page 3, 2013.
[2] Shidhartha Das, David Roberts, Seokwoo Lee, Sanjay Pant, David Blaauw, Todd Austin, Krisztián Flautner, and Trevor Mudge. A self-tuning dvs processor using delay-error detection and correction. *IEEE Journal of Solid-State Circuits*, 41(4):792–804, 2006.
[3] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.
[4] Dan Ernst, Shidhartha Das, Seokwoo Lee, David Blaauw, Todd Austin, Trevor Mudge, Nam Sung Kim, and Krisztián Flautner. Razor: circuit-level correction of timing errors for low-power operation. *IEEE Micro*, 24(6):10–20, 2004.

[5] Lulu Ge et al. Classification using hyperdimensional computing: A review. *IEEE Circuits and Systems Magazine*, 2020.

[6] Peter Hazucha, T Karnik, J Maiz, S Walstra, B Bloechel, J Tschanz, G Dermer, S Hareland, P Armstrong, and S Borkar. Neutron soft error rate measurements in a 90-nm cmos process and scaling trends in sram from 0.25-/spl mu/m to 90-nm generation. In *IEEE International Electron Devices Meeting 2003*, pages 21–5. IEEE, 2003.

[7] Michael Hersche, Sara Sangalli, Luca Benini, and Abbas Rahimi. Evolvable hyperdimensional computing: Unsupervised regeneration of associative memory to recover faulty components. In *2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pages 281–285. IEEE, 2020.

[8] Mohsen Imani, Samuel Bosch, Sohum Datta, Sharadhi Ramakrishna, Sahand Salamat, Jan M Rabaey, and Tajana Rosing. Quanthd: A quantization framework for hyperdimensional computing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(10):2268–2278, 2019.

[9] Mohsen Imani, Deqian Kong, Abbas Rahimi, and Tajana Rosing. Voicehd: Hyperdimensional computing for efficient speech recognition. In *2017 IEEE International Conference on Rebooting Computing (ICRC)*, pages 1–8. IEEE, 2017.

[10] JEDEC Standard JESD218. Solid-state drive (ssd) requirements and endurance test method. *Arlington, VA, JEDEC Solid State Technology Association*, 1:1–1, 2010.

[11] Xun Jiao, Vahideh Akhlaghi, Yu Jiang, and Rajesh K Gupta. Energy-efficient neural networks using approximate computation reuse. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1223–1228. IEEE, 2018.

[12] Xun Jiao, Mulong Luo, Jeng-Hau Lin, and Rajesh K Gupta. An assessment of vulnerability of hardware neural networks to dynamic voltage and temperature variations. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 945–950. IEEE, 2017.

[13] Xun Jiao, Dongning Ma, Wanli Chang, and Yu Jiang. Levax: An input-aware learning-based error model of voltage-scaled functional units. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(12):5032–5041, 2020.

[14] Xun Jiao, Dongning Ma, Wanli Chang, and Yu Jiang. Tevot: timing error modeling of functional units under dynamic voltage and temperature variations. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.

[15] Xun Jiao, Abbas Rahimi, Yu Jiang, Jianguo Wang, Hamed Fatemi, Jose Pineda De Gyvez, and Rajesh K Gupta. Clim: A cross-level workload-aware timing error prediction model for functional units. *IEEE Transactions on Computers*, 67(6):771–783, 2017.

[16] Pentti Kanerva. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive computation*, 1(2):139–159, 2009.

[17] Geethan Karunaratne, Manuel Le Gallo, Giovanni Cherubini, Luca Benini, Abbas Rahimi, and Abu Sebastian. In-memory hyperdimensional computing. *Nature Electronics*, 3(6):327–337, 2020.

[18] Geethan Karunaratne, Manuel Schmuck, Manuel Le Gallo, Giovanni Cherubini, Luca Benini, Abu Sebastian, and Abbas Rahimi. Robust high-dimensional memory-augmented neural networks. *Nature communications*, 12(1):1–12, 2021.

[19] Sung Kim, Patrick Howe, Thierry Moreau, Armin Alaghi, Luis Ceze, and Visvesh Sathe. Matic: Learning around errors for efficient low-voltage neural network accelerators. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6. IEEE, 2018.

[20] Guanpeng Li, Siva Kumar Sastry Hari, Michael Sullivan, Timothy Tsai, Karthik Pattabiraman, Joel Emer, and Stephen W Keckler. 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*, pages 1–12, 2017.

[21] Haitong Li, Tony F Wu, Abbas Rahimi, Kai-Shin Li, Miles Rusch, Chang-Hsien Lin, Juo-Luen Hsu, Mohamed M Sabry, S Burc Eryilmaz, Joon Sohn, et al. Hyperdimensional computing with 3d vrram in-memory kernels: Device-architecture co-design for energy-efficient, error-resilient language recognition. In *2016 IEEE International Electron Devices Meeting (IEDM)*, pages 16–1. IEEE, 2016.

[22] Jeng-Hau Lin, Xun Jiao, Mulong Luo, Zhuowen Tu, and Rajesh K Gupta. Vulnerability of hardware neural networks to dynamic operation point variations. *IEEE Design & Test*, 2020.

[23] Dongning Ma, Jianmin Guo, Yu Jiang, and Xun Jiao. Hdtest: Differential fuzz testing of brain-inspired hyperdimensional computing. In *Design Automation Conference (DAC)*, 2021.

[24] Dongning Ma and Xun Jiao. Molehd: Automated drug discovery using brain-inspired hyperdimensional computing, 2021.

[25] Dongning Ma, Rahul Thapa, Xingjian Wang, Cong Hao, and Xun Jiao. Workload-aware approximate computing configuration. In *Conference on Design, Automation & Test in Europe*. European Design and Automation Association, 2021.

[26] Anton Mitrokhin, P Sutor, Cornelia Fermüller, and Yiannis Aloimonos. Learning sensorimotor control with neuromorphic sensors: Toward hyperdimensional active perception. *Science Robotics*, 4(30), 2019.

[27] Abbas Rahimi, Sohum Datta, Denis Kleyko, Edward Paxon Frady, Bruno Olshausen, Pentti Kanerva, and Jan M Rabaey. High-dimensional computing as a nanoscalable paradigm. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 64(9):2508–2521, 2017.

[28] Abbas Rahimi et al. Hyperdimensional biosignal processing: A case study for emg-based hand gesture recognition. In *ICRC*, 2016.

[29] Abbas Rahimi, Pentti Kanerva, Luca Benini, and Jan M Rabaey. Efficient biosignal processing using hyperdimensional computing: Network templates for combined learning and classification of exg signals. *Proceedings of the IEEE*, 107(1):123–143, 2018.

[30] Abbas Rahimi, Pentti Kanerva, and Jan M Rabaey. A robust and energy-efficient classifier using brain-inspired hyperdimensional computing. In *Proceedings of the 2016 International Symposium on Low Power Electronics and Design*, pages 64–69, 2016.

[31] Brian Randell, Pete Lee, and Philip C. Treleaven. Reliability issues in computing system design. *ACM Computing Surveys (CSUR)*, 10(2):123–165, 1978.

[32] Brandon Reagen, Udit Gupta, Lillian Pentecost, Paul Whatmough, Sae Kyu Lee, Niamh Mulholland, David Brooks, and Gu-Yeon Wei. Ares: A framework for quantifying the resilience of deep neural networks. In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2018.

[33] Brandon Reagen, Paul Whatmough, Robert Adolf, Saketh Rama, Hyunkwang Lee, Sae Kyu Lee, José Miguel Hernández-Lobato, Gu-Yeon Wei, and David Brooks. Minerva: Enabling low-power, highly-accurate deep neural network accelerators. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pages 267–278. IEEE, 2016.

[34] Mohammad Samragh, Mohammad Ghasemzadeh, and Farinaz Koushanfar. Customizing neural networks for efficient fpga implementation. In *2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 85–92. IEEE, 2017.

[35] Manuel Schmuck, Luca Benini, and Abbas Rahimi. Hardware optimizations of dense binary hyperdimensional computing: Rematerialization of hypervectors, binarized bundling, and combinational associative memory. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 15(4):1–25, 2019.

[36] Rahul Thapa, Bikal Lamichhane, Dongning Ma, and Xun Jiao. Spamhd: Efficient text spam detection using brain-inspired hyperdimensional computing. In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2021.

[37] Rahul Thapa, Dongning Ma, and Xun Jiao. Hdxplore: Automated differential testing of brain-inspired hyperdimensional computing. In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2021.

[38] Ruixuan Wang, Fanxin Kong, Hasshi Sudler, and Xun Jiao. Hdad: Hyperdimensional computing-based anomaly detection for automotive sensor attacks. In *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2021.

[39] Tony F Wu, Haitong Li, Ping-Chen Huang, Abbas Rahimi, Jan M Rabaey, H-S Philip Wong, Max M Shulaker, and Subhasish Mitra. Brain-inspired computing exploiting carbon nanotube fets and resistive ram: Hyperdimensional computing case study. In *2018 IEEE International Solid-State Circuits Conference-(ISSCC)*, pages 492–494. IEEE, 2018.

[40] Fangfang Yang and Shaolei Ren. On the vulnerability of hyperdimensional computing-based classifiers to adversarial attacks. In *International Conference on Network and System Security*, pages 371–387. Springer, 2020.

[41] Jeff Zhang, Kartheek Rangineni, Zahra Ghodsi, and Siddharth Garg. Thundervolt: enabling aggressive voltage underscaling and timing error resilience for energy efficient deep learning accelerators. In *Proceedings of the 55th Annual Design Automation Conference*, pages 1–6, 2018.