Error Resilient Hyperdimensional Computing Using Hypervector Encoding and Cross-Clustering

Mohamed Mejri, Chandramouli Amarnath and Abhijit Chatterjee
School of Electrical and Computer Engineering
Georgia Institute of Technology, Atlanta, Georgia 30332–0250

2024 IEEE 42nd VLSI Test Symposium (VTS) | 979-8-3503-6378-4/24/$31.00 ©2024 IEEE | DOI: 10.1109/VTS60656.2024.10538955

***Abstract*—Emerging brain-inspired hyperdimensional computing (HDC) algorithms are vulnerable to timing and soft errors in associative memory used to store high-dimensional data representations. Such errors can significantly degrade HDC performance. A key challenge is *error correction* after an error in computation is detected. This work presents two novel error resilience frameworks for hyperdimensional computing systems. The first, called the *checksum hypervector encoding (CHE) framework*, relies on creation of a single additional hypervector that is a checksum of all the class hypervectors of the HDC system. For error resilience, element-wise validation of the checksum property is performed and those elements across *all* class vectors for which the property fails are removed from consideration. For an HDC system with K class hypervectors of dimension D, the second *cross-hypervector clustering (CHC) framework* clusters D, K-dimensional vectors consisting of the i-th element of each of the K HDC class hypervectors, $1 \leq i \leq K$. Statistical properties of these vector clusters are checked prior to each hypervector query and *all the elements of all K-dimensional vectors* corresponding to statistical outlier vectors are removed as before. The choice of which framework to use is dictated by the complexity of the dataset to classify. Up to three orders of magnitude better resilience to errors than the state-of-the-art across multiple HDC high-dimensional encoding (representation) systems is demonstrated.** [1]

***Index Terms*—Error resilience, Hyperdimensional computing**

I. Introduction

Hyperdimensional Computing (HDC) is an emerging learning paradigm that leverages the capabilities of high dimensional vectors to represent complex information. It achieves high efficiency (i.e, power-to-performance) in comparison to Deep Neural Networks (DNNs) in various domains [1], [2]. Modern edge hardware paradigms such as Field Programmable Gate Arrays (FPGAs) [3] are often used to make HDC systems more efficient. However, these hardware systems can suffer from memory access errors (bit-flips in accessed data) in the associative memory used to store the HDC model. Relevant soft errors are made worse by high energy particles [4]. Further, voltage scaling can lead to incorrect logic values being clocked into circuit flip-flops in associated logic. Due to their high dimensional symbolic representation, HDC systems are inherently robust to limited degrees of noise and error [5]. However, recent research indicates that HDC model accuracy declines substantially if the error rate exceeds 0.001% [6]. Traditional failure tolerance methods (e.g, triple modular redundancy (TMR) [7]) can be used to protect the HDC system against these kind of errors. However, these techniques are often expensive and require high degrees of redundancy (e.g, x3 times for TMR).

In this paper, we propose two novel error resilience techniques designed to mitigate the effects of soft errors (see Figure 1) in the associative memory (static random access memory, SRAM) of hyperdimensional computing systems. The first method consists of a *checksum hypervector encoding (CHE)* of the class hypervectors of the HDC system which are fetched from memory during class inference operations. Checksums, unlike parity checks, have been employed for error detection in fault-tolerant matrix computations and deep neural networks [8], [9] but their use for error correction in HDC systems has not been explored. [10] showed that checksums are able to detect errors but fails correcting them. In this paper, a novel error correction method is implemented by removing the faulty elements of concerned hypervectors. The second technique relies on *cross-hypervector clustering (CHC)* for error detection. Here, vectors corresponding to the i'th element of each class hypervector are clustered. Deviations from nominal cluster statistics of hypervector elements accessed from memory are used to detect errors. A novel correction mechanism is implemented by eliminating consideration the erroneous vector elements of all hypervectors corresponding to erroneous clusters. The two error resilience techniques can be generalized to broad HDC classification frameworks. Both techniques provide up to three orders of magnitude better error resilience compared to the state-of-the-art [11] and are used selectively depending on the HDC end application and use methodology.

In the following, prior work in error resilient hyperdimensional computing is discussed. In Section III, hyperdimensional computing preliminaries are introduced and an overview of the two error resilience techniques is given. In Section IV, the proposed error resilience mechanisms are described. In Section V, experimental evaluation of the two error resilience methods on state-of-the-art datasets is presented and compared against the current state-of-the-art. This is followed by conclusions in Section VI.

[1]Our codes and data are available at https://github.com/mmejri3/er-hdc

979-8-3503-6378-4/24/$31.00 ©2024 IEEE

II. Prior Work

Recent work on hyperdimensional computing (HDC) [12], [13] has focused on its algorithmic accuracy and energy efficiency [5]. There has been prior work on error resilience of deep neural networks [9], [14]. A broad theme has been to apply techniques from algorithm based fault tolerance [8], [15] to error tolerance in the linear matrix-vector computations of neural networks that feed the inputs to the activation functions of neurons in each layer of the network. In hyperdimensional computing, the class hypervectors are fetched from memory repeatedly for classification purposes rendering them vulnerable to memory access errors. The resilience of HDC algorithms to timing and soft errors in relevant associative memory was explored in [6], [16]. Single bit-flips were used to model errors in the hypervector elements of the HDC system [17]. Scaling and clamping of the hypervector elements to correct the effects of bit-flip errors on the hypervectors was introduced in [11]. This improved classification accuracy of the underlying HDC systems by four orders of magnitude over systems without any embedded error resilience. In contrast to this prior work, we propose two error resilience methods that provide up to 100X or 1000X times better error resilience than the state of the art [11]. Further, we study error detection and correction under a stronger error model that includes the effects of multiple bit-flips in the hypervector elements.

III. HDC Background

Hyperdimensional computing is composed of *hypervector encoding*, specialized training, and inference. These are described below.

A. Hypervector encoding

Hyperdimensional computing aims to map a signal sample (i.e, datapoint) from an N-dimensional space (feature space) to a D-dimensional space (symbolic space), where D $\gg$ N. It exploits the correlations among data points within a high-dimensional symbolic vector representation.

Record based encoding: In record based encoding, each datapoint (e.g, image, voice, ect) is represented by N features (e.g, pixels, frequencies, ect). the position of these features and the value that could take are encoded in hypervectors. Every feature position (e.g, the i^{th} pixel of images or i^{th}frequency of an M-frequency coefficient of the image or the spectrum, is assigned a unique channel ID, generated as a random binary hypervector, ensuring orthogonality. The possible values that the feature could have (e,g 255 for pixels) is divided into M bins, with a binary hypervector called *level hypervector* assigned to each bin. The hypervector datapoint is hence $S = hv^1 * ID^1 + hv^2 * ID^2 + ... + hv^n * ID^n$ *Kernel based encoding:* Kernel based encoding is discussed in [18]. It maps the datapoints x to a high dimensional space. The encoded hypervector $\phi(x) = cos(x.B + b).sin(x.B)$ where B is a matrix composed of normally distributed columns orthogonal to each other (i.e, $\delta(B_i, B_j) = 0 \; for \; i \neq j$), δ denotes the cosine similarity and b is a randomly generated hypervector that follows the uniform distribution ($b \sim \mathcal{U}[0, 2\pi]$).

B. Training and inference: Hyperdimensional computing

During the learning phase, HDC recognizes recurring patterns and avoids over-saturation of *class hypervectors* (one per class) in single-pass training [5]. Rather than simply combining all encoded information, this technique adjusts the contribution of each encoded data point to the class hypervectors based on the novelty it brings. If a data point is already present in a class hypervector, HDC adds little or no data to the model to avert hypervector saturation. If the prediction aligns with the anticipated outcome, no modifications are made to prevent overfitting. This adaptive updating process prioritizes and assigns greater weight to uncommon patterns in the final model representation. Suppose we have a new training data point, $\mathcal{H}$. HDC calculates the cosine similarity between H and all class hypervectors, C_s. The similarity of this data point with class i is computed as: $\delta_i = \delta(\mathcal{H}, C_i)$. Rather than naively incorporating the data point into the model, HDC updates the model according to the δ similarity. If the input data has a label l that accurately corresponds to the class, the model is updated as: $C_l \leftarrow C_l + \eta_1(1 - \delta_l) \times \mathcal{H}$, where η_1 refers to the learning rate. If the similarity between the class hypervector and the training hypervector is large (i.e, $\delta_l \approx 1$) the algorithm retains a small part of the training hypervector. If the input $l^{'}$ is misclassified or very similar to the wrong class hypervector (i.e, $\delta_{l'} \approx 1$), we add the hypervector to the correct class hypervector and subtract a portion (i.e, $\delta_{l'}$) of it from the wrong class hypervector as: $C_{l'} \leftarrow C_{l'} - \eta_2(\delta_{l'}) \times \mathcal{H}$.

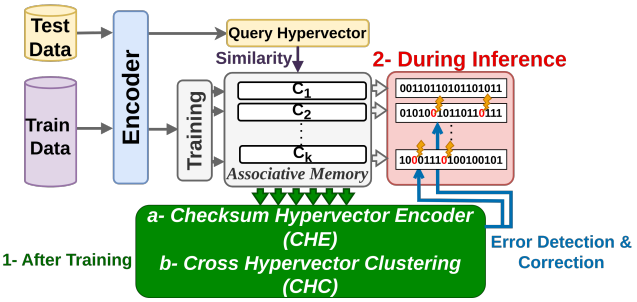


Fig. 1: Hyperdimensional computing framework and error resilience.

During inference, a *query hypervector* is generated from the input dataset using the encoding principles as described above and mapped to a class hypervector using the cosine similarity metric which determines the class to which the query belongs.

IV. Error resilience methodology

In this section, we present the error model, followed by the checksum (CHE) and clustering based (CHC) error resilience techniques.

A. Error model

We model errors in memory accesses of class hypervectors using single and multiple bit-flips [19] in the numerical (binary) representations of the hypervector elements. Such *soft errors* are assumed to be caused by radiation (a single alpha

particle strike can cause multiple bit upsets also referred as single event multiple-bit upsets [20]), cosmic rays, and timing errors such as induced by voltage scaling, signal coupling or power and ground bounce [21].

B. Checksum hypervector encoding (CHE)

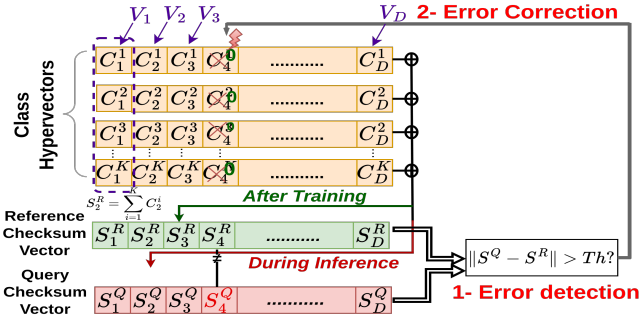


Fig. 2: Checksum hypervector encoding (CHE) mechanism

In Checksum Hypervector Encoding (CHE) (see Fig. 2), a *reference checksum hypervector* is constructed according to the equation $S_i^R = \sum_{j=0}^{K} C_i^j$, where C_i^j is the i^{th} element of the j^{th} class hypervector (out of K class hypervectors) and S_i^R is the i^{th} element of the *reference checksum hypervector*. During inference, a query checksum hypervector S^Q is computed using all the class hypervectors fetched from memory. This is compared against the reference checksum hypervector for error detection. If the difference between the reference and query checksum hypervectors is above a predefined threshold Th a detection flag is raised.

$$E_i = \begin{cases} 1, & \text{if } \|S_i^R - S_i^Q\|_1 > Th \quad (1) \\ 0, & \text{otherwise} \quad (2) \end{cases}$$

Empirically, we choose the threshold Th above (same for all E_i, $1 \leq i \leq D$, where D = hypervector dimension), such that the HDC system suffers from no more than an accuracy degradation of 1% before an error detection flag is raised. We assume that S^R is subject to error since it is stored in associative memory. The checksum hypervector encoding can detect errors in the class hypervector or the reference checksum hypervector. The reference hypervector is fetched from memory 3 times (as in TMR or triple modular redundancy [7]) and voting is performed to determine an error-free reference hypervector. Without TMR above, it is seen that the probability of checksum aliasing is very small for systems designed to minimize memory access overhead.

CHE Error Correction mechanism: For correction, if $E_i \neq 0$, then the $i'th$ element of *all the class hypervectors* is set to 0, $1 \leq i \leq D$, and inference is performed with the resulting modified class hypervectors. In addition to applying TMR to the reference checksum hypervector, we also restrict the range of its element values, clipping all values in excess of the maximum range to the maximum or minimum allowed (determined from training data). The reference checksum S^R has the overhead of an additional hypervector with dimension D represented in N bitwise as the class hypervectors.

C. Cross hypervector clustering (CHC)

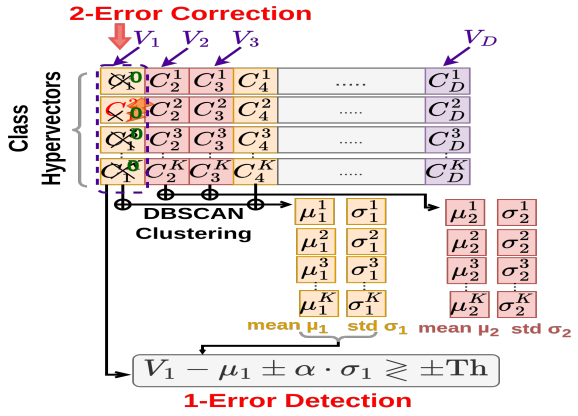


Fig. 3: Cross hypervector clustering (CHC) mechanism

Figure 3 describes the steps of the clustering based error detection method. K-dimensional vectors (K = number of class hypervectors) V_i, $1 \leq i \leq D$ called *cross-class vectors*, consisting of the elements V_i = [C_i^1, C_i^2, ..., C_i^K] (K class hypervectors) are constructed. These vectors are clustered using the DBSCAN [22] model. DBSCAN [22] identifies clusters based on the density of data points, efficiently distinguishing between core points, border points, and noise, thereby enabling the detection of clusters of varied shapes and densities. The clustering algorithm's minimum distance, denoted by ϵ, is contingent on the selected encoder, the utilized dataset, and the intended number of clusters. For every cluster, the mean (μ) and the variance (σ^2) of the cross-class vectors is computed and stored. During inference, if the vector $V_i - \mu - \alpha.\sigma > Th$ or if $V_i - \mu + \alpha.\sigma < -Th$, where Th is a calibrated threshold for the cluster to which V_i belongs, then a detection flag is raised.

$$E_i = \begin{cases} 1, & \text{if } \|V_i - \mu\| > Th + \alpha.\sigma \quad (3) \\ 0, & \text{otherwise} \quad (4) \end{cases}$$

In case an error is detected, corrective action is performed as described below.

CHC Error Correction mechanism: Error correction is performed as follows. If $E_i = 0$, then correction is performed by setting [C_i^1, C_i^2, ..., C_i^K] = [0, 0, ..., 0]. However, this approach may produce false positives; multiple such elements in a cluster might be erroneously deemed divergent from the reference and, thus, mislabeled as defective. Such misclassification can cause a marginal decrease in accuracy (less than 1%) in error-free models. The chosen error detection threshold, Th, is intrinsically linked to the minimum distance, ϵ_{min}, set during clustering of the cross-class vectors V_i. A larger ϵ_{min} results in fewer clusters, increasing the average distance between centroids and their members. To maintain consistent accuracy, a corresponding increase in Th is required. Nevertheless, a higher Th reduces error resilience but increases accuracy of HDC models. Associative memory is used to store the mean and standard deviation values of each cluster. Let

	n	k	Train size	Test size	Description
GTSRB	3072	43	39209	12630	Image Classification
UCIHAR	561	6	6213	1554	Activity Recognition (Mobile)
ISOLET	617	26	6238	1559	Voice Recognition
Fashion-MNIST	784	10	60000	10000	Image Classification
CIFAR100	3072	100	50000	10000	Image Classification

TABLE I: DATASETS (n: Feature size, k: Number of classes)

Cl be the number of clusters and K the number of class hypervectors. The memory overhead thus includes the mean (μ) and variance (σ) stored as class hypervector bitwidth N data. The reference data is protected via TMR (i.e, 3 copies are stored for error resilience), increasing corresponding memory overhead.

V. Experimental Validation

A. Experimental Setup

The proposed error resilience methods were evaluated on benchmark datasets (Fashion-MNIST, GTSRB, ISOLET and UCIHAR), as presented in Table I. Class hypervector clustering and the checksum calculations are performed offline on a CPU (11th Gen Intel® Core™ i7-11800H @ 2.30GHz) and the error resilience assessment is performed on both CPU and FPGA. For the FPGA validation platform, the error resilience model was first synthesized using Xilinx Vitis High-Level Synthesis. The power consumption of the error resilience model was determined using the Xilinx Vivado XPower tool. The FPGA employed for validation was the Xilinx Zynq UltraScale+ MPSoC ZCU104. The three encoders (n-gram, record and kernel, discussed in Section III) were implemented on CPU. For FPGA testing we use only the kernel encoder [18] as it yields the best HDC performance. In the following experiments, the HDC has a hypervector dimension of $D = 10^4$ in CPU simulations and $D = 1000$ in FPGA hardware validation. When assessed on CPU, the error rates are between 10^{-8} and 10^{-2} and from 10^{-8} and 10^{-1} when assessed on the FPGA. Each experiment is repeated five times, and the mean of the experimental results for each experiment is presented.

B. Error resilience and overhead comparison

In this section, we compare the performance and overhead of the checksum hypervector encoding (CHE) and the cross hypervector clustering (CHC) methods on benchmark datasets (ISOLET, CIFAR100, Fashion-MNIST and GTSRB).

Figure 6(a) shows the *failure error rate* of CHC and CHE on several datasets. The *failure error rate* is defined as the error rate that causes a 5% accuracy drop from nominal accuracy. The nominal accuracy of each dataset is: 91.4% for UCIHAR, 91.4% for UCIHAR, 93.3% for GTSRB, 84.9% for Fashion-MNIST and 40% for CIFAR100, An observed trend is that error resilience increases with higher HDC performance. This is due to the error correction technique (removing erroneous hypervector indices) artificially reducing the dimensionality of the class hypervectors (dropping dimensions). As seen in Figure 6(b), this dimensionality reduction has a greater effect on complex datasets, reducing accuracy faster as hypervector indices are set to zero.

In Figure 6(b), the 5% of accuracy loss corresponding to failure error rate is attained with a lower proportion of cross-class vectors suppressed dimensions when the nominal accuracy is low (CIFAR100 and FashionMNIST datasets). Figure 6(a) further shows that CHE exhibits superior error resilience when nominal accuracy is low (CIFAR100 and Fashion-MNIST cases), while CHC is more robust to errors when the nominal accuracy is high (UCIHAR and GTSRB cases). In the CHC scheme, error resilience is affected by two parameters: the number of clusters and the detection threshold. Enhancing error resilience requires lowering the detection threshold to identify erroneous outlier cross-class vectors. However, a lower threshold may trigger false alarms, thereby increasing the cross-class vectors suppression rate as shown in figure 6(b), and consequently the accuracy loss, particularly for datasets with low nominal accuracy. Apart from the CIFAR100 scenario, which lowers the error resilience of CHE, the resilience of CHE remains unchanged.

The CHC technique is seen to over-suppress cross-class vectors. This approach proves effective only when the classification task can tolerate aggressive cross-class vectors pruning, which is typically evident in high-accuracy classification tasks. By contrast, the CHE technique enhances error resilience by selectively suppressing only the erroneous cross-class vectors. However, errors within the reference checksum vector can impair error detection in CHE.

Table II shows the relative memory overhead of CHC compared to CHE. We observe that for datasets with a lower number of classes (i.e, Fashion-MNIST and UCIHAR) CHC has lower memory overhead than the CHE. CHC has higher overhead for datasets with a higher number of classes (i.e, GTSRB and CIFAR100). This is because the memory overhead of the clustering technique is proportional to the number of the dataset classes. Conversely, the overhead from the checksum remains constant across a range of datasets or class numbers, necessitating only a single checksum hypervector of fixed dimension.

	GTSRB	CIFAR100	Fashion-MNIST	UCIHAR
Relative Overhead of CHC (CHE=1X)	3.78	6.79	0.45	0.29

TABLE II: Relative Memory Overhead of CHC to CHE (CHE=30KB)

C. Accuracy Results

In this experiment we assess the error resilience of checksum hypervector encoding (CHE) and cross hypervector clustering (CHC), and compare them to an HDC system with no error resilience systems (no-resilience baseline) and Scale+Clip [11] using three different encoding systems (Ngram-based encoding [23], record-based encoding [23] and kernel based encoding [18], discussed in Section III). The resilience to errors of a system is defined in relation to the baseline model as follows: If the error rate of model failure, as previously defined, is $10^{-\alpha}$, and the error rate of failure for the baseline model is $10^{-\beta}$, then the error resilience can be expressed as $\frac{10^{-\alpha}}{10^{-\beta}} = 10^{\beta-\alpha}$. We also compare the proposed schemes against the Scale+clip prior work of [11].

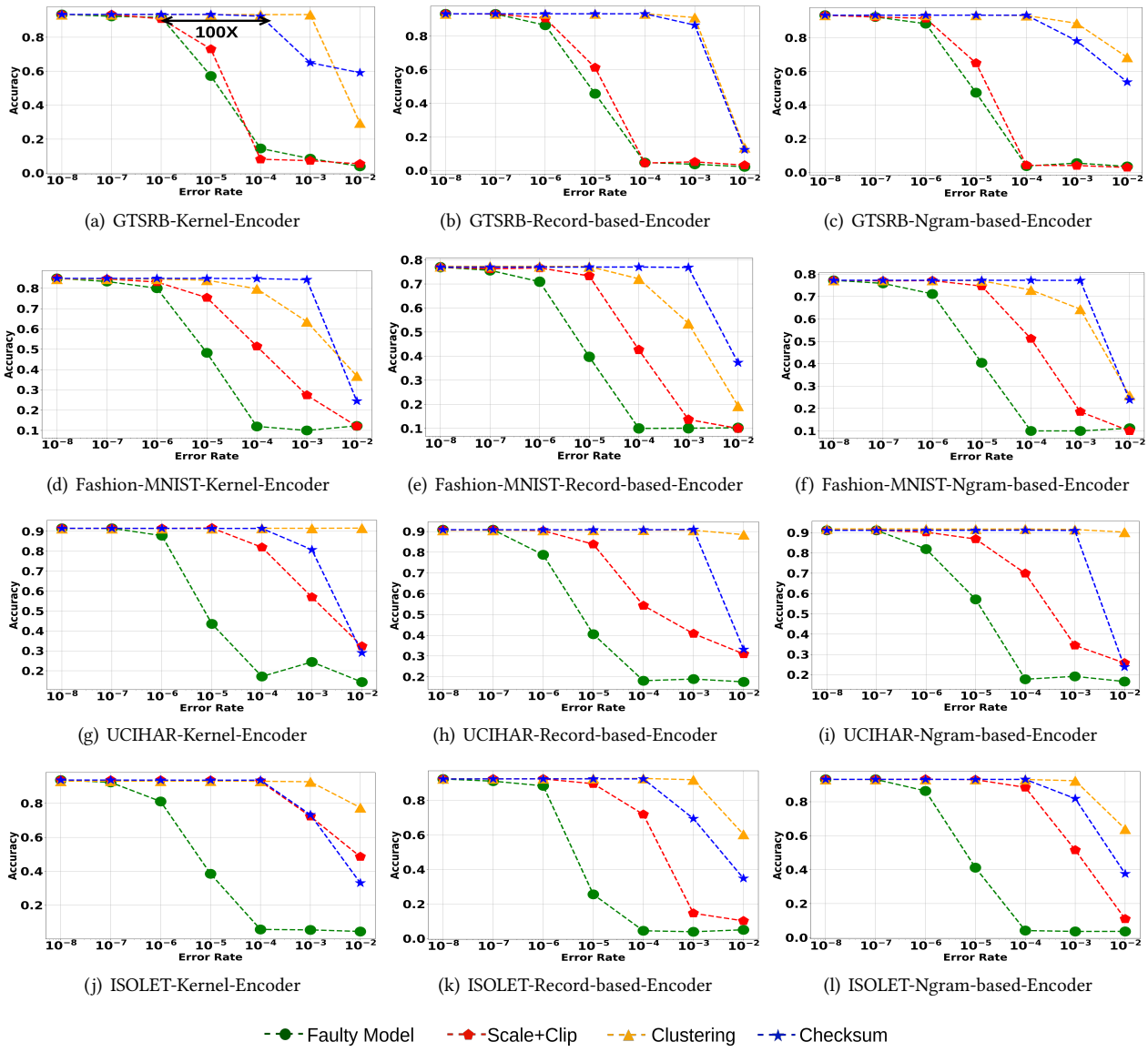


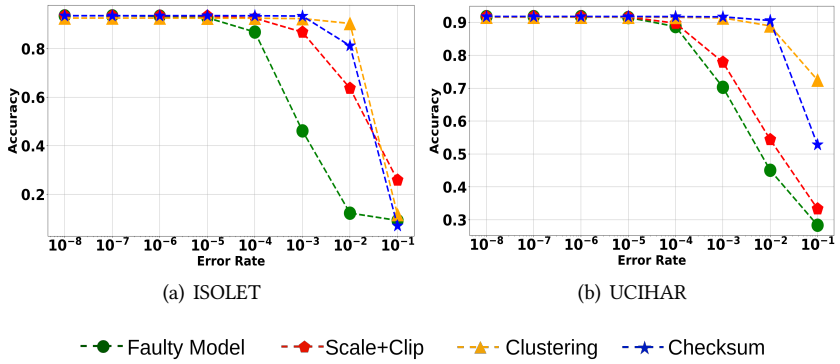
Fig. 4: Error resilience in HDC test cases using CHC, CHE, Scale+Clip [11] and the case of no resilience method used (Faulty Model). The 100X in Fig. (a) indicates the extent of improvement in error resilience (two orders of magnitude increase in error rate before performance drop).

As an overall observation the CHE and CHC based method outperforms the state-of-the-art (i.e, the Scale+Clip [11]) in terms of error resilience for all three encoding systems for increasing error rates from 10^{-7} to 10^{-2}.

For example in Figure 4(g), the accuracy of HDC on UCIHAR using the kernel-based encoder and Scale+Clip [11] drops at 10^{-4} while the same accuracy of CHC and the CHE based methods drop after 10^{-3}. A similar dynamic is seen for Figure 4(h) and Figure 4(i) where the checksum and clustering assessed on UCIHAR are at least three orders of magnitude more robust to errors than Scale+Clip [11] (i.e, the error resilience of the proposed technique is at least 1000 higher than Scale+Clip [11]). From Figures4(a), 4(c) and 4(b) we observe higher error resilience using the clustering (CHC) technique than the checksum (CHE) technique, and both methods provide greater error resilience than the baselines. From Figures 4(j), 4(k) and 4(l) the HDC model with CHC provides up to three orders of magnitude more error resilience than Scale+Clip [11] when assessed on ISOLET. When assessed on Fashion-MNIST, CHC achieves one order of magnitude more error resilience than Scale+Clip [11] regardless of the type of encoder. In this scenario, CHE is at least three orders of magnitude more error resilient than the state-of-the-art. The clustering based method achieves better error resilience than the checksum based method when assessed on UCIHAR and ISOLET and GTSRB and less robustness to errors when tested on Fashion-MNIST. *We thus see that improvements in error resilience depend on dataset complexity and the HDC encoder used.*

D. Hardware Validation

We compare the CHE and CHC methods to the state of the art [11] on an FPGA test platform. The reference checksum

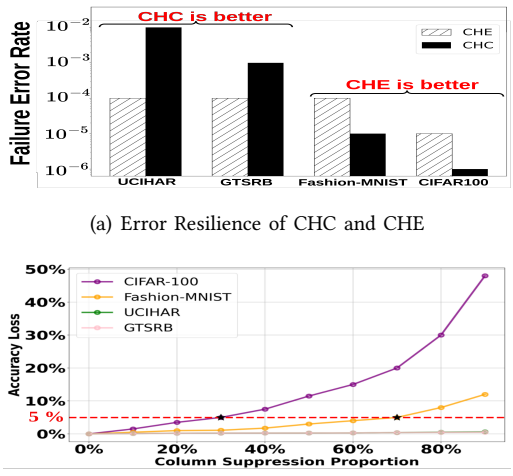


(a) ISOLET (b) UCIHAR

Fig. 5: Accuracy of different error resilience methods on FPGA platform (i.e, Xilinx ZCU104)

		Utilization Overhead				Inference Time (ms)	Power (W)	Energy (mJ)	Error resilience
		FF	LUT	DSP	BRAM				
ISOLET	Baseline model	27022	51317	63	261	30.7	0.921	28.27	
	Scale+clip	27437	52033	65	309	31.2	0.951	29.67	1X
	Checksum (CHE)	33458	55664	67	311	30.9	1.026	31.7	10X
	Clustering (CHC)	29457	55213	63	333	31	0.997	30.9	100X
UCIHAR	Baseline model	27004	51302	62	175	24.3	0.884	21.48	
	Scale+clip	27403	52002	64	187	24.4	0.899	21.93	1X
	Checksum (CHE)	29060	52930	66	189	24.3	0.92	22.37	100X
	Clustering (CHC)	28816	53273	62	194	24.3	0.920	22.37	100X

TABLE III: Hardware Overhead: CHC & CHE Vs state-of-the-art error resilience methods



(a) Error Resilience of CHC and CHE

(b) Cross-vector-removal effect on HDC Accuracy

Fig. 6: Error Resilience of CHC and CHE and Cross-vector-removal effect on HDC

hypervector for CHE, and the mean and variance clustering vectors for CHC were stored using BRAMs. The checksum and clustering vectors as well as the class hypervector were subjected to multiple bit flip error injection (See Section IV-A). The indices of the bit flips are randomly selected. Table III summarizes the overhead (resource utilization, inference time and power consumption) of each method. In terms of resource utilization, CHE and CHC have higher overhead than Scale+Clip [11] and the baseline model. CHE incurs less memory overhead than CHC. As such, the relative overhead figures are as expected (when referring to Section V-B) for ISOLET (due to its higher number of classes) but unexpected for UCIHAR (which has a lower number of classes). One explanation is that individual clusters (i.e, mean and variance) will take individual BRAM cells (each BRAM is able to store up to 18 kb of data). A large number of low dimension vectors (i.e, CHC means and variances) need more BRAM cells than a single high dimensional vector (reference checksum hypervector). However, the checksum technique uses more DSPs than the clustering method since it involves an additional operation (adding the class hypervectors together).

Both CHC and CHE have comparable inference time (less than 1% difference), higher than the baseline model and Scale+Clip. CHE consumes more energy than CHC. Figure 5 shows the error resilience of the checksum (CHE) and clustering (CHC) based method the ISOLET and UCIHAR datasets. For ISOLET, CHC provides two orders of magnitude more error resilience than Scale+Clip [11] while CHE provided 10X more error resilience than the baseline. For UCIHAR, the checksum and the clustering methods provide two orders of magnitude more error resilience than the state-of-the-art.

VI. Conclusion

This work presented two error resilience methods: the checksum hypervector encoding (CHE) and the cross-hypervector clustering (CHC). Both methods enable up to three orders of magnitudes of improvement in error resilience compared to the state-of-the-art. CHE provides an almost constant level of error resilience with low memory overhead, while CHC enhances error resilience depending on the classification performance of the hyperdimensional computing with higher memory overhead.

Acknowledgment

This research was supported by NSF Grant No. 2128149 and by the School of ECE at Georgia Institute of Technology.

References

[1] Fatemeh Asgarinejad, Anthony Thomas, and Tajana Rosing. Detection of epileptic seizures from surface eeg using hyperdimensional computing. In *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pages 536–540. IEEE, 2020.

[2] Peer Neubert, Stefan Schubert, and Peter Protzel. An introduction to hyperdimensional computing for robotics. *KI-Künstliche Intelligenz*, 33:319–330, 2019.

[3] 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.

[4] Alfio Di Mauro, Francesco Conti, Pasquale Davide Schiavone, Davide Rossi, and Luca Benini. Pushing on-chip memories beyond reliability boundaries in micropower machine learning applications. In *2019 IEEE International Electron Devices Meeting (IEDM)*, pages 30–4. IEEE, 2019.

[5] Alejandro Hernandez-Cane, Namiko Matsumoto, Eric Ping, and Mohsen Imani. Onlinehd: Robust, efficient, and single-pass online learning using hyperdimensional system. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 56–61. IEEE, 2021.

[6] Sizhe Zhang, Ruixuan Wang, Jeff Jun Zhang, Abbas Rahimi, and Xun Jiao. Assessing robustness of hyperdimensional computing against errors in associative memory. In *2021 IEEE 32nd International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 211–217. IEEE, 2021.

[7] Barry W Johnson. An introduction to the design and analysis of fault-tolerant systems. *Fault-tolerant computer system design*, 1:1–84, 1996.

[8] Kuang-Hua Huang and Jacob A Abraham. Algorithm-based fault tolerance for matrix operations. *IEEE transactions on computers*, 100(6):518–528, 1984.

[9] Chandramouli Amarnath, Mohamed Mejri, Kwondo Ma, and Abhijit Chatterjee. Soft error resilient deep learning systems using neuron gradient statistics. In *2022 IEEE 28th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, pages 1–7. IEEE, 2022.

[10] Cynthia J. Anfinson and Franklin T. Luk. A linear algebraic model of algorithm-based fault tolerance. *IEEE Transactions on Computers*, 37(12):1599–1604, 1988.

[11] Sizhe Zhang, Mohsen Imani, and Xun Jiao. Scalehd: robust brain-inspired hyperdimensional computing via adapative scaling. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, pages 1–9, 2022.

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

[13] 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.

[14] Elbruz Ozen and Alex Orailoglu. Just say zero: Containing critical bit-error propagation in deep neural networks with anomalous feature suppression. In *Proceedings of the 39th International Conference on Computer-Aided Design*, pages 1–9, 2020.

[15] Sujay Pandey, Suvadeep Banerjee, and Abhijit Chatterjee. Error resilient neuromorphic networks using checker neurons. In *2018 IEEE 24th International Symposium on On-Line Testing And Robust System Design (IOLTS)*, pages 135–138. IEEE, 2018.

[16] Dongning Ma, Sizhe Zhang, and Xun Jiao. Robust hyperdimensional computing against cyber attacks and hardware errors: A survey. In *Proceedings of the 28th Asia and South Pacific Design Automation Conference*, pages 598–605, 2023.

[17] 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 *Proceedings of the 55th Annual Design Automation Conference*, pages 1–6, 2018.

[18] Yang Ni, Nicholas Lesica, Fan-Gang Zeng, and Mohsen Imani. Neurally-inspired hyperdimensional classification for efficient and robust biosignal processing. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, pages 1–9, 2022.

[19] Jose Maiz, Scott Hareland, Kevin Zhang, and Patrick Armstrong. Characterization of multi-bit soft error events in advanced srams. In *IEEE International Electron Devices Meeting 2003*, pages 21–4. IEEE, 2003.

[20] Balkaran Gill, Michael Nicolaidis, and Chris Papachristou. Radiation induced single-word multiple-bit upsets correction in sram. In *11th IEEE International on-line testing symposium*, pages 266–271. IEEE, 2005.

[21] Weiwei Shan, Longxing Shi, and Jun Yang. In-situ timing monitor-based adaptive voltage scaling system for wide-voltage-range applications. *IEEE Access*, 5:15831–15838, 2017.

[22] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

[23] Mohsen Imani, Chenyu Huang, Deqian Kong, and Tajana Rosing. Hierarchical hyperdimensional computing for energy efficient classification. In *Proceedings of the 55th Annual Design Automation Conference*, pages 1–6, 2018.