ScaleHD: Robust Brain-Inspired Hyperdimensional Computing via Adapative Scaling

Sizhe Zhang
Villanova University

Mohsen Imani
UC Irvine

Xun Jiao*
Villanova University

ABSTRACT

Brain-inspired hyperdimensional computing (HDC) has demonstrated promising capability in various cognition tasks such as robotics, bio-medical signal analysis, and natural language processing. Compared to deep neural networks, HDC models show advantages such as light-weight model and one/few-shot learning capabilities, making it a promising alternative paradigm to traditional resource-demanding deep learning models particularly in edge devices with limited resources. Despite the growing popularity of HDC, the robustness of HDC models and the approaches to enhance HDC robustness has not been systematically analyzed and sufficiently examined. HDC relies on high-dimensional numerical vectors referred to as hypervectors (HV) to perform cognition tasks and the values inside the HVs are critical to the robustness of an HDC model. We propose **ScaleHD**, an adaptive scaling method that scales the value of HVs in the associative memory of an HDC model to enhance the robustness of HDC models. We propose three different modes of **ScaleHD** including *Global*-**ScaleHD**, *Class*-**ScaleHD**, and *(Class + Clip)*-**ScaleHD** which are based on different adaptive scaling strategies. Results show that **ScaleHD** is able to enhance HDC robustness against memory errors up to $10,000X$. Moreover, we leverage the enhanced HDC robustness in exchange for energy saving via voltage scaling method. Experimental results show that **ScaleHD** can reduce energy consumption on HDC memory system up to 72.2% with less than 1% accuracy loss.

1 INTRODUCTION

The developments in artificial intelligence (AI) and machine learning (ML), such as deep neural networks (DNNs), promise enormous societal and economic benefits. However, their deployment on hardware faces daunting difficulties due to their extremely-demanding and increasing computation requirements which may not always be satisfied by resource-constrained platforms. Recently, inspired by the way brain works with cognition tasks, especially the very large brain circuit size, hyperdimensional computing (HDC) has been introduced as an alternative computational model [7, 8, 17]. Compared to DNNs, HDC has shown several key advantages including more compact model, lower computation requirements, and one/few-shot learning capability, making it a promising AI paradigm in resource-constrained platforms such as mobile and wireless devices [2, 24]. Such versatility has led to its success in various application domains including robotics [16], natural language process [17, 20], drug discovery [14], and multimedia processing [5, 15].

As the transistor size scales down to deep nanometer era, microelectronic circuits are becoming increasingly susceptible to microelectronic variability [9], such as variations in manufacturing, operating conditions (e.g., voltage droops and temperature fluctuations), and aging. Such variations can cause delay uncertainty which can prevent circuits from meeting their timing specification, thus resulting in *timing errors*. Timing errors typically appear as faults (bit flips) in the high-level operations, which can cause incorrect computing results. Furthermore, the radiation-induced soft error rate per bit is also estimated to increase 8% with each technology scaling by semiconductor vendors [3]. Such variations pose imminent threat to the correctness and quality of results delivered by computing systems, which typically requires a near-perfect execution with extremely low error rates to guarantee correctness.

While AI/ML methods are generally more error-robust than conventional computing and the quality of model output results is relatively insensitive to the rising error rates, recent study shows that the inference accuracy of DNN models can still drop significantly when the error rates are too high, such as 0.001% [13, 21].

Similarly, while HDC is claimed to be robust to hardware errors due to its holographic distributed representation with information equally distributed into high-dimensional vectors, recent studies show that HDC also experiences a significant accuracy drop with error rate reaches 0.001% [23]. Considering the increasing reliability threat posed by microelectronic variations, especially with the burgeoning use of microelectronic devices in mobile and wireless applications that can constantly experience variations in operating conditions (e.g., supply voltage droops and temperature fluctuations), enhancing the HDC robustness to hardware errors are of utmost importance.

To address this problem, we present **ScaleHD** in this paper, which aims to improve the robustness of HDC models to bit flip errors, which is one typical error caused by microelectronic variations and/or soft errors. **ScaleHD** leverages the distribution characteristics of the values in the HDC model to perform adaptive scaling, specifically, on the class hypervectors to obtain enhanced robustness. The enhanced robustness further open the doors for opportunistic reduction of design margin (e.g., voltage) to increase the operational efficiency of hardware in HDC systems.

Our contributions are as follows:

- We analyze the distribution of values of the class HVs in HDC models across multiple applications, based on which we propose **ScaleHD**. **ScaleHD** is an adaptive scaling method that can re-distribute the values in class HVs in HDC to

*Xun Jiao is the corresponding author: xun.jiao@villanova.edu

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ICCAD '22, October 30-November 3, 2022, San Diego, CA, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9217-4/22/10…$15.00
https://doi.org/10.1145/3508352.3549376

improve the robustness against bit flip errors while incurring little to no impact on HDC inference accuracy.

- We propose three different modes of **ScaleHD**, including *Global*-**ScaleHD**, *Class*-**ScaleHD**, and *(Class + Clip)*-**ScaleHD**, each focusing on different levels of scaling to achieve different level of performance. Experimental results across various applications and HDC configurations show that **ScaleHD** can significantly enhance HDC robustness to memory errors up to $10,000X$.
- Using **ScaleHD**, we are able to enable higher energy efficiency. Specifically, we propose to use voltage scaling to reduce the memory energy. While voltage scaling can lead to memory errors, the enhanced robustness with **ScaleHD** can lead up to 72.2% energy saving on HDC system in 22nm SRAM architecture.

2 RELATED WORKS

AI/ML robustness to hardware errors have been intensively studied in the past, especially in the DNN accelerators context [11, 18, 19]. DNN robustness to timing errors caused by voltage and temperature variations has been first studied in [6], which shows that DNNs can experience significant accuracy drop for error rate as low as 0.001%, which is consistent with [21]. A set of DNN error injection tools are then proposed to assess the robustness to DNNs including Ares [18] and TensorFi [12]. A common observation of all these studies is that DNN is more robust to hardware errors to a certain degree depending on data types, values, data reuses, and types of layers in the design.

Brain-inspired HDC has been proposed as a light-weight novel computing paradigm that has shown promising performance in robotics [16], natural language process [17, 20], drug discovery [14], and multimedia processing [5, 15]. For example, HDC outperforms state-of-the-art DNN models (e.g., graph/recurrent neural networks) in drug discovery in terms of accuracy with a significantly reduced computing costs [14]. While HDC is claimed to be robust to hardware errors [4, 17], recent studies show that, like DNNs, HDC can also experience sharp accuracy drop with low error rates (e.g., 0.001%) [23]. Currently, most HDC studies rely on inherent error tolerance of existing HDC algorithms [4, 17]. This unfortunately may not be the best way to protect HDC from hardware errors especially considering the increasing threat from microelectronic variations and environmental uncertainty and. Extra hardware circuitry like Razor [23] may mitigate the impact of bit flip errors but cost extra energy and chip area [22].

Considering the algorithmic difference between DNNs and HDC, existing algorithm-level DNNs error enhancement techniques cannot be directly applied to HDC algorithm. In this paper, we propose a general algorithmic-level method that does not require any hardware modification.

3 HDC BACKGROUND

In this section, we describe the HDC model base element, operations and operating processes including the **Encoding**, **Training**, **Retraining**, and **Inference**. The flow shows in Fig. 1 can help understand the basic framework of HDC.

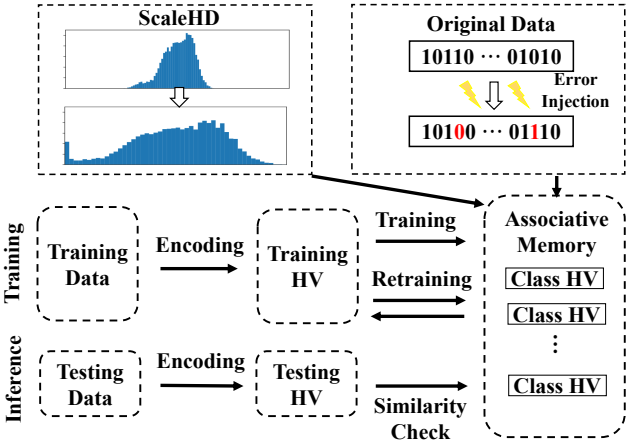


Figure 1: HDC with ScaleHD framework diagram

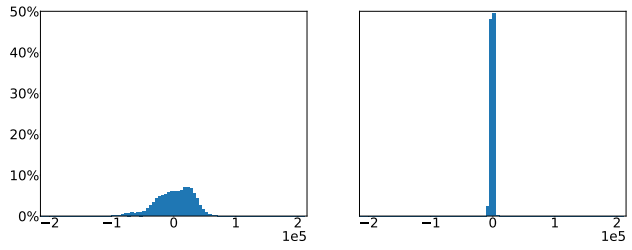
Basic Component and Operation HDC uses a high-dimensional (e.g., $D = 10000$) pseudo-random and holographic vector($\vec{H}$) called a hypervector(HV) as the basic element shown in Eq. 1. HV values can be integers, binary or bipolar. In HDC, HVs serve as a data block to represent information depending on the application, such as language characters, audio signal amplitudes, and image pixel values. As part of HDC, different HV operations are used to bundle and bind different information. Arithmetic operations supported by HV are addition, multiplication, and permutation. Additions and multiplications use two input HVs to perform element-by-element additions and multiplications. Permutation rotates one HV cyclically to generate a new HV. The dimensions of the input and the output HVs stay the same across all the operations. By multiplying and permuting, generated HVs are orthogonal to their original, while adding retains 50% of the information of each original HV.

$$\overrightarrow{H} = [h_1, h_2, \ldots, h_d] \tag{1}$$

Encoding HDC's framework begins with encoding. In HDC, samples are mapped from original data to HVs. Both training and inference are performed using the HVs. The encoding map raw features to HVs. For our HDC model, we employ the most common record-based encoding [2]. In the beginning, it creates position hypervectors($\vec{H_{pos}}$) and level hypervectors($\vec{H_{level}}$). They are all bipolar$\{-1, 1\}$ and have the same dimension. Each position HV is generated randomly and orthogonally to each other. Every level HV correlates with the nearby level HVs. The subsequent level HVs are generated by modifying certain number of bipolar elements from nearby level HV. Each feature in a sample will have its feature HV($\vec{H_f}$) generated by multiplying the corresponding level HV and position HV. The sample HV($\vec{H_s}$) will be generated by adding the feature HVs($\vec{H_f}$) in total. The encoding is based on the equation Eqn. 2.

$$\vec{H_f} = \vec{H_{level}} * \vec{H_{pos}} \qquad \vec{H_s} = \vec{H_{f_1}} + \vec{H_{f_2}} \ldots \vec{H_{f_n}} \tag{2}$$

Training The encoding stage encodes each input sample into a sample HV. The training stage simply add all the sample HVs ($\vec{H_s^n}$) of the same class together to create a class HV($\vec{A_n}$) representing that specific class, as shown in Eq. 3. This is an element-by-element accumulation. HDC framework stores all class HVs in the associative



(a) Class 6 HV (b) Class 7 HV

Figure 2: Original data distribution histogram in associative memory

memory.

$$\vec{A_n} = \sum \vec{H_s^n} \tag{3}$$

Retraining Furthermore, HDC retraining, other than the single-pass method, is optional, but can significantly increase the model accuracy through several epochs of retraining. In this first stage, the training data will be tested. Upon incorrect prediction, the query HV($\vec{H_s}$) will be added to the correct class HV($\vec{A}$) and subtracted from the wrong predicted class HV($\vec{A'}$). In this manner, the class HV can be enhanced from samples, and other class HVs that may be distorted will be reduced in relation to them. The following Eq. 4 summarizes the process. Moreover, the β learning rate can be an adaptive learning rate so as to enhance retraining.

$$\begin{aligned} \vec{A'} &= \vec{A'} - \beta\vec{H_s} \\ \vec{A} &= \vec{A} + \beta\vec{H_s} \end{aligned} \tag{4}$$

Inference The inference stage classifies unseen samples using the pre-trained class HVs in the associative memory. First, same as the training stage, testing samples are encoded into sample HVs (referred to as "query HV"). Second, the HDC model measures the similarity between the query HV($\vec{H_s}$) and every class HV($\vec{A_n}$) in the associative memory. The most typical similarity metric is cosine similarity as Eq. 6 [17]. Eventually, the class(l) with the highest similarity with the query HV will be used as the inference result of the sample as shown in Eq. 5.

$$l = argmax(\{\cos(\vec{H_s}, \vec{A_1}), \cos(\vec{H_s}, \vec{A_2}), \ldots, \cos(\vec{H_s}, \vec{A_n})\}) \tag{5}$$

4 CLASS HV VALUE DISTRIBUTION ANALYSIS

In this section, we present a key observation by profiling and analyzing the value distribution of class HVs in HDC models, which inspires the development of **ScaleHD**. After training HDC models on different applications, we profile the values of class HVs in HDC models and examine their value distribution. We find that for different applications, class HVs would have different value ranges and distributions, as shown in Table 1. Because each application has a different amount of samples, and different HDC framework use different encoding method and training strategy, we need to use different bit-width to implement the each element in class HV for different HDC system to completely store these models. For example, in our implementation, the value range is (-5266, 6340) for CARDIO after training, which means 16 bits (i.e., int16) is enough to store the model. While for HAR with a value range of (-155595, 104273), we need 32 bits (i.e., int32) to store the value.

Now we take a deeper look into HAR's class HVs values by profiling the value distribution, as shown in Fig. 2. We use two examples — "class 6" HV and "class 7" HV. As shown in Fig. 2, we can find the range and distribution of values for "class 6" HV and "class 7" HV are notably different. This is caused by HAR's unbalanced dataset. Class 6 HV value has a wider range because of more samples comparing to class 7. Class 7 HV values are largely clustered around 0 due to less training samples. From the Fig. 2 (a) and Table. 1, we realize the value range of this associative memory is larger than int16 (-32,768 to +32,767), but much less than int32 (-2,147,483,648 to +2,147,483,647), which means there are "empty" bits (i.e., "0" bits) in each value in the most significant positions.

Note that different value will have different relative error for a bit flip. For example, for a 4-bit unsigned system, if the original data is 0010 (i.e., 2_{10}) and there is a bit flip on the least significant bit (LSB), the relative error is $1/2 = 50\%$. However, if the original data is 1110 (i.e., 14_{10}), the relative error of the same bit flip is only $1/14 = 7\%$. This inspires us to consider the following question: can we scale up values in HDC model without affecting the HDC classification result? If so, we can reduce the relative effects of bit flips and hence enhance the HDC robustness through fully using these extra "space" comes with fixed data-width to reduce the relative error. This inspired us to develop **ScaleHD**.

5 APPROACH OF SCALEHD

In this section, we first provide theoretical foundations for **ScaleHD**. Then, we present the detailed process of **ScaleHD**, along with three different modes of **ScaleHD**: Global-**ScaleHD**, Class-**ScaleHD**, and (Class + Clip)-**ScaleHD**. Finally, we use special metric briefly evaluate the robustness enhancements made by **ScaleHD**.

5.1 Mathematical Proof of ScaleHD

Before we introduce **ScaleHD**, we provide a theoretical foundation for **ScaleHD** by demonstrating the equivalence of HDC functionality before and after **ScaleHD**. As we described in the section on HDC background, HDC uses HV as its basic component. Moreover, the HDC uses the cosine similarity between two vectors as the main part of the inference process. By using cosine similarity, we can compare the direction between two vectors, and linearly scaling up/down each value in the class HV with the same ratio will not impact the direction of vectors. Therefore, the results of the cosine similarity check will not change, resulting in the same inference result. A mathematical proof of this can also be provided by Eq. 6. Due to this fact, scaling the same ratio to each value in a class HV will not affect the inferences of the HDC system.

$$\begin{aligned} \cos(\alpha\vec{H_p}, \vec{H_q}) &= \frac{\alpha \cdot \vec{H_p} \cdot \vec{H_q}}{||\alpha \cdot \vec{H_p}|| \times ||\vec{H_q}||} \\ &= \frac{\sum_{i=1}^{d} \alpha \cdot h_{pi} \cdot h_{qi}}{\sqrt{\sum_{i=1}^{d} \alpha^2 \cdot {h_{pi}}^2} \cdot \sqrt{\sum_{i=1}^{d} {h_{qi}}^2}} \\ &= \frac{\sum_{i=1}^{d} h_{pi} \cdot h_{qi}}{\sqrt{\sum_{i=1}^{d} {h_{pi}}^2} \cdot \sqrt{\sum_{i=1}^{d} {h_{qi}}^2}} \\ &= \cos(\vec{H_p}, \vec{H_q}) \end{aligned} \tag{6}$$

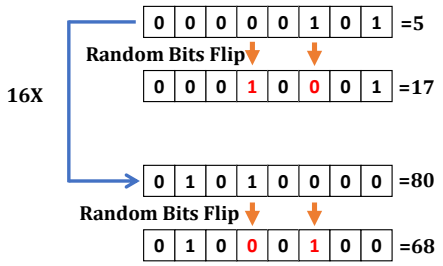
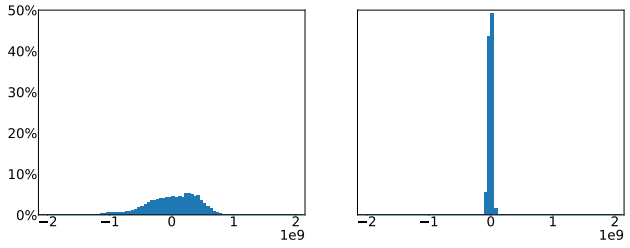


Figure 3: Mechanism of scaling to enhance robustness



(a) Class 6 HV (b) Class 7 HV

Figure 4: Data distribution histogram in associative memory with Global-ScaleHD

5.2 Mechanisms and Implementation of ScaleHD

ScaleHD is an set of methods can enhance the HDC robustness though value scaling. Essentially, it is a post-processing step for the HDC model (associative memory) after training and before inference. The process is specific to its data type, so it can be applied to every integer representation HDC model with a wide range of settings and applications. Furthermore, the process does not require any additional hardware as it is a software-based modification. There are limited overhead costs, and the process is almost free of charge. Following that, we will discuss the details of each **ScaleHD**.

Global-ScaleHD : The first mode of **ScaleHD** is Global-**ScaleHD**. It basically scales up all values in class HVs with the same ratio. Specifically, we first let $\overrightarrow{H} = [V_1, V_2, \ldots, V_d]$ be a d dimensional class HV, and $\mathbf{A} = \{H_1, H_2, \ldots, H_n\}$ be an associative memory storing n class HVs. w represent the data type the system use. In this paper, we choice common used INT32, INT16 and INT8 as our data type. Second, we find the maximum absolute value $\mathbf{V_a}$ in the $\mathbf{A}$. We also define the maximum value that can be represented by the current bit width as $\mathbf{V_n}$. For example, for a 32-bit int32 value, the maximum value $\mathbf{V_n}$ is 2,147,483,647. Then, we compute the ratio α between $\mathbf{V_n}$ and $\mathbf{V_a}$. Last, by scaling up every value in $\mathbf{A}$ by α times, all values in $\mathbf{A}$ get scaled up to $\mathbf{A_s}$ without overflow and precision loss. This process can be summarized as Algorithm 1. Scaled value will have less relatively error comparing to the original value when same bit flips occur. Fig. 3 shows a example how scale can reduce relatively error against bit flip. The original value is 5, after bit flip, it become to 17. The relative error is $|17-5|/5 = 240\%$. After 16 times scale up, the value become 80, With the same random bit flip, the relative error is $|68-80|/80 = 15\%$ which is much smaller. Fig. 4 shows the value distribution of associative memory after Global-**ScaleHD**. We can find though the distribution pattern seems the same, the x-axis is 10000X larger after Global-**ScaleHD**. This will enhance the robustness of HDC.

$$\alpha = \frac{V_n}{V_a} \qquad \mathbf{A_s} = A \times \alpha \tag{7}$$

Algorithm 1 Global-**ScaleHD**

Parameters: Associate Memory A; Scaled Associate Memory A_s; Target data-width w.
Allocate memory for scaled associate memory according to original associate memory A and target-width w
$A_s = malloc(A, w)$
Find Max absolute value in the associate memory
$V_{max} \leftarrow max(A)$
$V_{min} \leftarrow min(A)$
$V_a \leftarrow max(abs(V_{max}), abs(V_{min}))$
$V_n \leftarrow max(w)$ # Find the max value of this data type
Compute scaling ratio
$\alpha \leftarrow V_n / V_a$
Scale and save each value in the associate memory to the scaled associate memory
$A_s \leftarrow \alpha * A$
return A_s

Class-ScaleHD : Next, we present the second mode Class-**ScaleHD**. This is inspired by our observation that even for the same HDC model, different class HVs would have different value distributions. Thus, instead of scaling up class HVs using a global ratio, we individually scale up each class HV as shown in Fig. 5. Note that according to Section 5.1, the class-specific scaling up will also not change the classification results. We present the following steps to implement Class-**ScaleHD**, which is also illustrated in Algorithm 2. First, we find the maximum absolute value $\mathbf{V_c}$ for each class HV $\mathbf{H_c}$. Then, we compute a ratio α between $\mathbf{V_n}$ (the maximum value that can be represented by w) and $\mathbf{V_c}$. Third, we scale up this class HV by α. This process can be simplified as Eq. 8. We repeat these steps for each class HV so that we can get an updated associative memory. Fig. 6 shows the value distribution of associative memory after Class-**ScaleHD**. Compare to Fig. 4, we can find the Class-**ScaleHD** is effective especially for unbalanced data. Because of HDC's special training strategy, the class with more samples usually lead to a larger absolute value of class HV. Global-**ScaleHD** can not handle this situation. The different distribution of "class 6" HV and "class 7" HV validates this idea. Because the value range of each class HV is likely to be different from the whole associative memory value range, this method can fully scale up with balanced data trained associative memory which will further enhance the HDC robustness.

$$\alpha = \frac{V_n}{V_c} \quad \mathbf{H_s} = H \times \alpha \quad \mathbf{A_s} = \{H_{s1}, H_{s2}, \ldots, H_{sn}\} \tag{8}$$

(Class + Clip)-ScaleHD : At the end, we present the third mode of **ScaleHD** — (Class + Clip)-**ScaleHD**. The key idea is to cut/clip some extreme values in the class HV value distribution so that we are able to scale up even more, hence providing more robustness to HDC model. We propose the following steps. First, we use Class-**ScaleHD** to the model to fully scale up the value. Second, we set

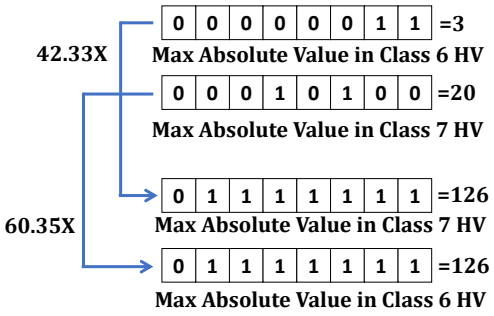
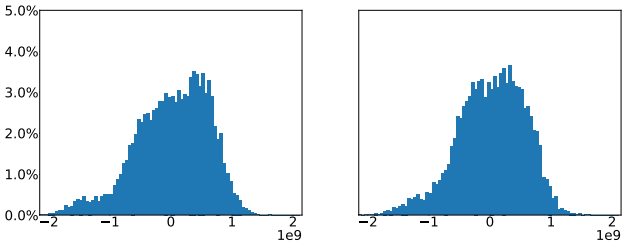


Figure 5: Class-ScaleHD

Algorithm 2 Class-**ScaleHD**

Parameters: Associate Memory A; Scaled Associate Memory A_s; Target data-width w.
Allocate memory for scaled associate memory according to original associate memory A and target-width w
$A_s = malloc(A, w)$
# Iterate each class HV H_c in the Associate Memory A
for $H_c \in A$ **do**
    # Find Max absolute value in the class HV
    $V_{max} \leftarrow max(H_c)$
    $V_{min} \leftarrow min(H_c)$
    $V_a \leftarrow max(abs(V_{max}), abs(V_{min}))$
    $V_n \leftarrow Maxvalueforw$
    # Compute scaling ratio
    $\alpha \leftarrow V_n / V_a$
    # Scale and save each value in the class HV to the target scaled class HV
    $H_s \leftarrow \alpha * H_c$
end for
return A_s



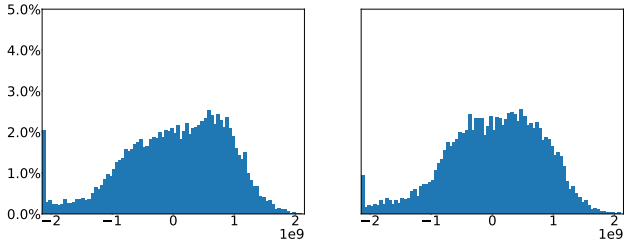
(a) Class 6 HV (b) Class 7 HV

Figure 6: Data distribution histogram in associative memory with Class-ScaleHD

an absolute clip value C, and thus we have two clip points at "+C" and "-C". Third, (Class + Clip)-**ScaleHD** will clip every value if it is larger/smaller than the clipping values to reduce the range of values in class HVs. The principle of clip is shown in Eq. 9. Fig. 7 shows the value distribution of associative memory after clipping, and we can find some value has been clipped on both ends of the distribution. After the clipping, we perform the Global-**ScaleHD** like scaling based on Eq. 7. The value distribution further "fill up" the possible range that can be represented by current bit width after clip and scale. This detail of the procedure shows in Algorithm 3.

Table 1: Pre-trained associative memory value range, selected clip ratio and absolute clip value for 32-bit.

Dataset	Minimum	Maximum	Clip Ratio(R)	Clip Value (C)
CARDIO	-5266	6340	0.605	1.3e+9
HAR	-155595	104273	0.698	1.5e+9
ISOLET	-22070	28294	0.652	1.4e+9
MNIST	-958416	778792	0.466	1e+9



(a) Class 6 HV (b) Class 7 HV

Figure 7: Data distribution histogram in associative memory with (Class + Clip)-ScaleHD

Please note that because clipping has changed certain values in the class HVs, the class HV will be affected. As a result, the classification results and the inference results will be impacted. If we want more scaling space to provide more robustness, the absolute clip value should smaller. But with a smaller absolute clip value means more values will get clipped and the accuracy may drop more. To explore the trade-off between clipping values and HDC inference accuracy, we performed experiment for clipping value v.s. accuracy in 32-bit to find a balance between robustness and accuracy. The result is shown in Fig. 8. Through a careful examination, we choose a absolute clip value for each application so that the inference accuracy loss is less than 1%. Next, We calculate the ratio between the selected clip value and 32-bit data range as the clip ratio for each application. We use this ratio for other data-width (Class+Clip)-**ScaleHD** experiment. The clip ratio(R) and absolute clip value(C) (after Class-**ScaleHD**) for int32 we choose in our experiment are in Table. 1.

$$HV[i] = \begin{cases} +C, & HV[i] > +C \\ HV[i], & -C \le HV[i] \le +C \\ -C, & HV[i] < -C \end{cases} \quad (9)$$

5.3 Robustness Enhancement Evaluation

As an initial step in assessing the ability of the **ScaleHD** to enhance HDC's robustness before exam with testing dataset, we explore an evaluation metric specially for HDC to evaluation the **ScaleHD** enhancement. Given that HDC uses cosine similarity for inference, we use the cosine similarity between error inject model and original model as the evaluation metric to check how error influenced HDC model get protected by **ScaleHD**. In general, the higher the cosine similarity, the more likely the error inject model will be

Algorithm 3 (Class+Clip)-**ScaleHD**

Parameters: Associate Memory A; Scaled Associate Memory A_s; Target data-width w, Clip ratio for this application R
Allocate memory for scaled associate memory according to original associate memory A and target-width w
$A_s = malloc(A, w)$
# Iterate each class HV H_c in the Associate Memory A
for $H_c \in A$ **do**
    # Find Max absolute value in the class HV
    $V_{max} \leftarrow max(H_c)$
    $V_{min} \leftarrow min(H_c)$
    $V_a \leftarrow max(abs(V_{max}), abs(V_{min}))$
    $V_n \leftarrow Maxvalue for w$
    # Compute scaling ratio
    $\alpha \leftarrow V_n / V_a$
    # Scale and save each value in the class HV to the target scaled class HV
    $H_s \leftarrow \alpha * H_c$
end for
# Compute clip value for this data-width
$C \leftarrow V_n * R$
for $H \in A_s$ **do**
    if H > C **then**
        $H \leftarrow C$ # Clip positive extreme values
    end if
    if H < -C **then**
        $H \leftarrow -C$ # Clip negative extreme values
    end if
end for
# Scale after clip
$A_s = V_n / C * A_s$
return A_s

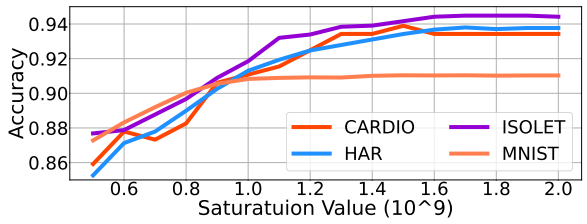


Figure 8: Absolute clip values influence accuracy across different applications (32-bit with (Class + Clip) -ScaleHD).

close to the original model, which further demonstrates how robust the model is. We run a short test on four applications, each with a different error rate, through 32bit settings. The results of Fig. 9 indicate that, although HDC is robust by itself, **ScaleHD** is superior at overcoming random bit-flip errors. Furthermore, Class-**ScaleHD** has performed significantly better than Global-**ScaleHD**, while (Class+Clip)-**ScaleHD** has demonstrated the strongest performance in this evaluation metric.

Table 2: Error-free accuracy for different datasets and configuration.

Configuration	ISOLET	HAR	CARDIO	MNIST
32-bit	94.42%	93.77%	93.43%	91.04%
16-bit	94.42%	93.77%	93.43%	91.04%
8-bit	94.42%	91.71%	92.96%	91.01%

6 EXPERIMENTAL RESULTS

To fully explore the effectiveness of **ScaleHD**, we test HDC with **ScaleHD** on four different applications with three different data-width. Datasets including speech recognition(**ISOLET** [1]), human activities (**HAR** [2]), medical diagnosis(**CARDIO** [3]) and image Classification(**MNIST** [10]). In our model development, we implement a basic approach of HDC model without special encoding method and training strategy which means our model accuracy might not meet state of art HDC model accuracy. But experiment still effectively show the effectiveness robustness enhancement of **ScaleHD** to general HDC model. Hardware errors are injected to the HDC models as random bit flips with a pre-defined error rate, similar to existing studies [11, 18, 19, 22]. This error model randomly flips a single bit upon the occurrence of an error in memory. We inject errors to HDC's associative memory with aimed error rates and error models during the inference stage. We apply **ScaleHD** to the HDC's associative memory after the training stage and before the inference stage.

6.1 Robustness Enhancement by ScaleHD

We use three different HDC configurations in terms of three different data width: 32-bit, 16-bit, and 8-bit across four different applications. As shown in Table. 1, the value range of class HVs is different across different applicaiton. We perform Global-**ScaleHD** as the baseline quantization method for our 16/8-bit experiment for HAR and MNIST, and 8-bit experiment for ISOLET and CARDIO in order to reduce the precision loss caused by fixed ratio quantization. The different HDC configurations and their corresponding (error-free) accuracy are presented in Table 2. Note that quantizing to 8-bit data width incur negligible accuracy loss. Nevertheless, we focus on the relative accuracy drop after injecting errors and relative robustness enhancement.

For each configuration, we sweep 90 error rates equally space on a logarithmic scale from 10^{-10} to 1, e.g., $8 * 10^{-10}$, $9 * 10^{-10}$, $1 * 10^{-9}$ etc. At each error rate, we repeat our experiment 10 times and report the mean accuracy. The overall experimental results are shown in Fig. 10, based on which we present several key observations.

First, by applying **ScaleHD**, HDC robustness is significantly enhanced across all application datasets. For example, in ISOLET 32-bit (Fig. 10 (a)), the baseline classifier starts to see accuracy drop at error rates 10^{-7}, while after applying (Class + Clip)-**ScaleHD**, it can push error rates to 10^{-3} without accuracy loss. This is $10,000X$

[1] http://archive.ics.uci.edu/ml/datasets/ISOLET
[2] https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones
[3] https://archive.ics.uci.edu/ml/datasets/cardiotocography

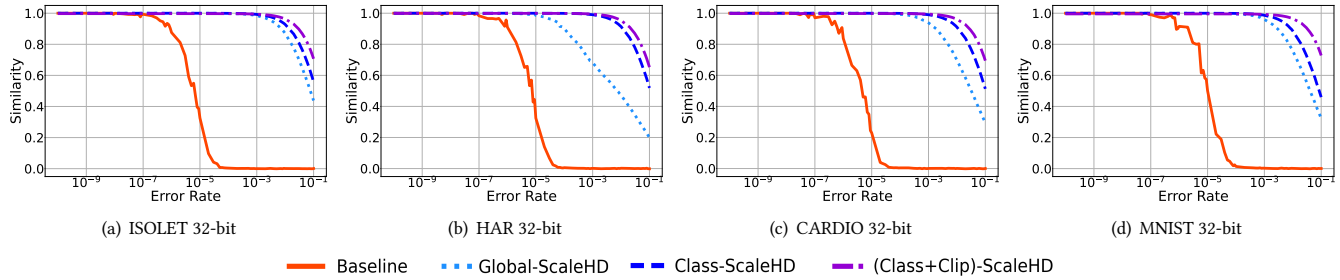


Figure 9: Evaluation ScaleHD with cosine similarity check

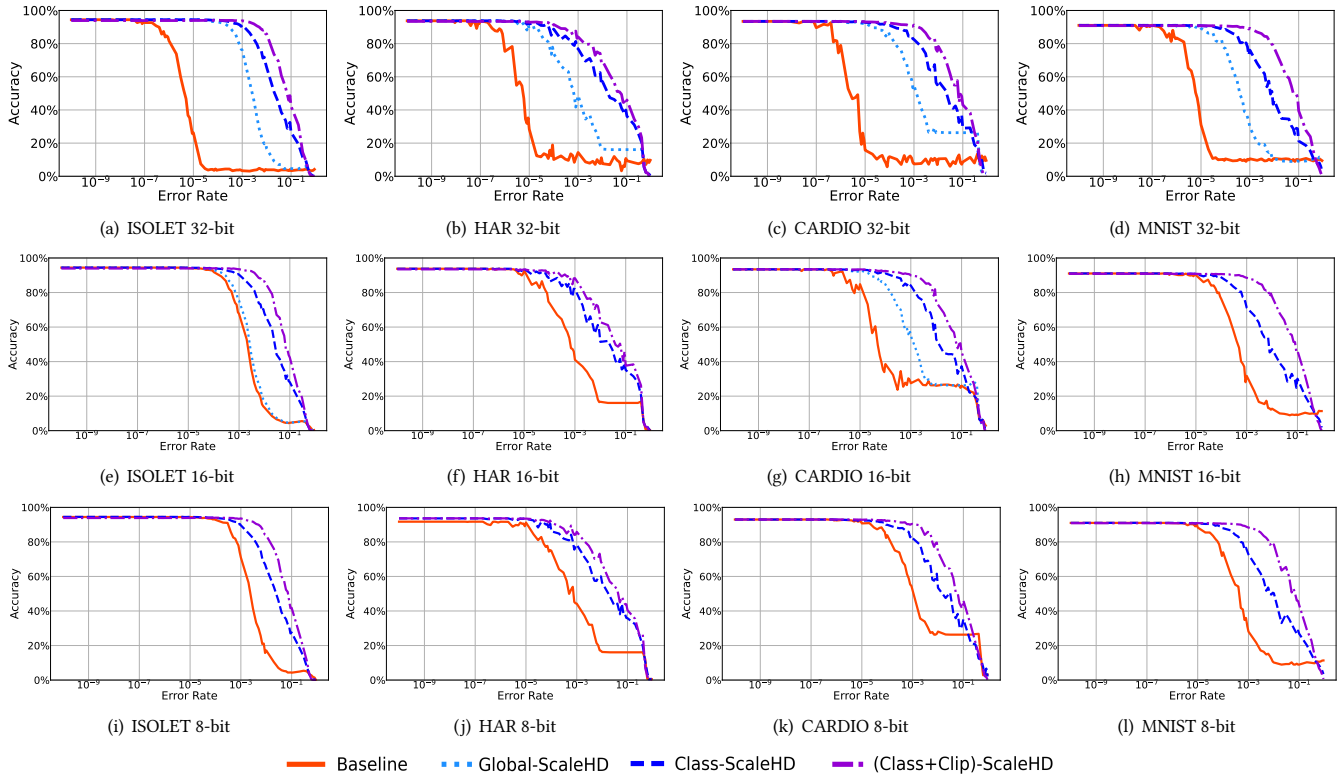


Figure 10: ScaleHD effects on HDC Robustness

robustness improvement. Similarly, for all the other 32-bit HDC classifiers (HAR, CARDIO, MNIST), i.e., Fig. 10 (b) - (d), the robustness are also significantly improved with 1000X - 10000X.

Second, among several modes of **ScaleHD**, we can consistently see that (Class + Clip)-**ScaleHD** outperforms Class-**ScaleHD**, which outperforms Global-**ScaleHD**. Using 32-bit MNIST (Fig. 10 (d)) as our example, we can see that (Class + Clip)-**ScaleHD** can push the error rates to around 10^{-3} without incurring accuracy loss, while for Global-**ScaleHD** and Class-**ScaleHD**, the corresponding metrics are around 10^{-5} and 10^{-4}. This suggests that, for this application, (Class + Clip)-**ScaleHD** can improve robustness 10X and 100X more than Class-**ScaleHD** and Global-**ScaleHD**, respectively. Similar phenomenon can be observed for all the application datasets and configurations across Fig. 10 (a) - (l).

For certain scenarios, as we mentioned, we use Global-**ScaleHD** as model quantization method for the baseline. For example, in 8-bit ISOLET (Fig. 10 (i)), the value range of class HVs are already beyond the range that can be represented by 8 bits. That is, all the bit positions are "full", and there are no "empty" bits for scaling. Actually, in this case, we need to down-scale the HV values to accommodate each value within 8 bits. Therefore, Global-**ScaleHD** can be seen as the baseline and quantization method in this case as we mention in the experiment setting. Regardless, by applying Class-**ScaleHD** and (Class + Clip) -**ScaleHD**, we are still able to improve HDC robustness around 100X.

Another interesting observation is in HAR 8-bit (Fig. 10 (j)) experiment. The accuracy of the model is higher with Class-**ScaleHD** (93.61%) and (Class+Clip)-**ScaleHD** (93.64%) compared to the baseline (91.71%). The situation is unusual as we proved that the error-free model with Global scale and Class scale will not affect the model inference accuracy, while the (Class+Clip)-**ScaleHD** may only reduce less

Table 3: Energy saving through voltage scaling across different datasets and settings

	Baseline			Global **ScaleHD**			Class **ScaleHD**			(Clip+Class) **ScaleHD**		
	32-bit	16-bit	8-bit	32-bit	16-bit	8-bit	32-bit	16-bit	8-bit	32-bit	16-bit	8-bit
CARDIO	48.39%	48.73%	61.22%	61.49%	59.15%	61.22%	64.20%	63.21%	64.20%	66.82%	66.54%	68.65%
HAR	47.01%	59.15%	61.49%	59.73%	59.15%	61.49%	63.21%	61.49%	65.43%	65.86%	66.23%	67.06%
ISOLET	48.00%	48.00%	65.86%	66.23%	65.86%	65.86%	68.65%	68.65%	68.65%	72.20%	72.20%	71.97%
MNIST	51.02%	61.49%	61.22%	59.73%	61.49%	61.22%	64.20%	64.90%	64.90%	69.57%	70.70%	70.21%
Average	48.60%	54.34%	62.45%	61.79%	61.41%	62.45%	65.07%	64.56%	65.80%	68.61%	68.92%	69.47%

than 1% accuracy in our setting. In spite of the fact that this phenomenon does not manifest itself in other experiments, we believe it remains reasonable. First of all, the 8-bit model is quantized from the 32-bit model, and thereby the precision of the model may suffer in the quantization(integer linear scaling down). As shown in the table, HAR, CARDIO, and MNIST all experience a decline in baseline accuracy with a 8-bit model. Furthermore, the ratio of scaling down may also influence model precision. When the ratio of scaling down is high, greater precision is likely to be lost. (Class+Clip)-**ScaleHD** may result in a lower precision loss than the Global- since (Class+Clip)- **ScaleHD** need less quantization scaling down ratio. Consequently, the Class-**ScaleHD** and (Class+Clip)- **ScaleHD** are also effective in recovering the precision loss from the quantization in some circumstances.

Last but not least, if we compare Fig. 10 (a)(b)(c) with Fig. 9, we find that the two plots follow similar trends. This shows despite only comparing cosine similarity between the original model and the error inject model, it is still able to predict how errors affect the HDC model's accuracy without putting the dataset under test. Therefore, these results show our evaluation metric is a reliable tool for evaluating HDC model robustness.

6.2 Energy Saving by ScaleHD

Recent research indicates that HDC associative memory can save up to 72.5% energy with an additional razor circuit to prevent errors [22]. To see how the robustness enhancement from **ScaleHD** can benefit HDC system, we performed simulations based on the relationship between voltage scaling and bit error rates, as depicted in Eq. 10 obtained from a real SRAM measurement fabricated in FDX22 (22nm) technology [1]. We map the tolerable error rates to the equation to obtain the corresponding voltage scaling budget. Then, using $P = CV^2f$, we can obtain the corresponding energy saving with less than 1% accuracy loss. This refers to [22]'s setting. Additionally, we carried out our simulation on each of the four applications, using a variety of data types. Furthermore, we examined the average energy savings across the four applications using the same settings. The details of energy savings can be found in Table 3. As for 16-bit ISOLET, CARDIO, and all 8-bit experiments, we employ Global-**ScaleHD** as the quantization method. We put this part of results both in the baseline and Global-**ScaleHD** section in Table 3.

Note that since HDC has inherent error tolerance, even the baseline HDC without quantization can save around 50% energy with negligible accuracy loss(<1%). Additionally, by using the Global-**ScaleHD**, the Class-**ScaleHD**, and the (Class + Clip)-**ScaleHD**, the average energy savings will rise to 62%, 65%, and 69%, respectively. Some settings within ISOLET and MNIST even able to save more than 70% of energy. When compared to the results found in [22] (Up to 72.5% energy saving), the memory has the capability of achieving the same level of energy savings with the **ScaleHD** applied. Nonetheless, existing work [22] requires extra protection circuit for the memory, which costs more space ,energy and require customized hardware, but **ScaleHD** is purely a software approach with no modification to the hardware and no overhead except a one-time post-processing procedure which takes place after training and prior to inference. Based upon our analysis, we believe our proposal is superior to the existing work.

$$Bit\ Error\ Rate = 2e{+}08\ *\ e^{-61.69*V_{dd}} \tag{10}$$

7 CONCLUSION

In this paper, we propose **ScaleHD**, an adaptive scaling method that scales the value of HV in associative memory to enhance the robustness of HDC models. We propose three different modes of **ScaleHD** including Global-**ScaleHD**, Class-**ScaleHD**, and (Class + Clip)-**ScaleHD**. We evaluate **ScaleHD** by performing error injection experiments with a wide range of error rates, datasets, HDC configurations. Results show that **ScaleHD** is able to enhance HDC robustness against memory errors up to 10000X. Moreover, we leverage the enhanced HDC robustness in exchange for energy-saving via the voltage scaling method. Experimental results show that **ScaleHD** can reduce energy consumption on HDC memory system up to 72.2% with less than 1% accuracy loss.

ACKNOWLEDGMENTS

The authors would like to thank Ruixuan Wang and Dongning Ma from DETAIL lab at Villanova University for source code related to error injection. This work was partially supported by Villanova Faculty Summer Grant and NSF grant #2028889. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] 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*, pages 30–4. IEEE, 2019.

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

[3] 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.

[4] 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.

[5] 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.

[6] 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.

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

[8] 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.

[9] Jaydeep P Kulkarni, Carlos Tokunaga, Paolo A Aseron, Trang Nguyen, Charles Augustine, James W Tschanz, and Vivek De. A 409 gops/w adaptive and resilient domino register file in 22 nm tri-gate cmos featuring in-situ timing margin and error detection for tolerance to within-die variation, voltage droop, temperature and aging. *IEEE Journal of Solid-State Circuits*, 51(1):117–129, 2015.

[10] Yann LeCun et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[11] 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.

[12] Guanpeng Li, Karthik Pattabiraman, and Nathan DeBardeleben. Tensorfi: A configurable fault injector for tensorflow applications. In *2018 IEEE International symposium on software reliability engineering workshops (ISSREW)*, pages 313–320. IEEE, 2018.

[13] 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.

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

[15] Alec Xavier Manabat et al. Performance analysis of hyperdimensional computing for character recognition. In *ISMAC*, 2019.

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

[17] 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.

[18] 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.

[19] David Stutz, Nandhini Chandramoorthy, Matthias Hein, and Bernt Schiele. Bit error robustness for energy-efficient dnn accelerators. *Proceedings of Machine Learning and Systems*, 3, 2021.

[20] 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.

[21] 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.

[22] Sizhe Zhang, Ruixuan Wang, Dongning Ma, Jeff Zhang, Xunzhao Yin, and Xun Jiao. Energy-efficient brain-inspired hyperdimensional computing using voltage scaling. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2022.

[23] 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.

[24] Zhuowen Zou, Yeseong Kim, Farhad Imani, Haleh Alimohamadi, Rosario Cammarota, and Mohsen Imani. Scalable edge-based hyperdimensional learning system with brain-like neural adaptation. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–15, 2021.