Testing and Enhancing Adversarial Robustness of Hyperdimensional Computing

Dongning Ma, *Student Member*, IEEE, Tajana Šimunić Rosing, *Fellow*, IEEE, Xun Jiao, *Member*, IEEE

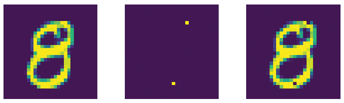
***Abstract*—Brain-inspired hyperdimensional computing (HDC), also known as Vector Symbolic Architecture (VSA), is an emerging "non-von Neumann" computing scheme that imitates human brain functions to process information or perform learning tasks using abstract and high-dimensional patterns. Compared with deep neural networks (DNNs), HDC shows advantages such as compact model size, energy efficiency, and few-shot learning. Despite of those advantages, one under-investigated area of HDC is the adversarial robustness; existing works have shown that HDC is vulnerable to adversarial attacks where attackers can add minor perturbations onto the original inputs to "fool" HDC models, producing wrong predictions. In this paper, we systematically study the adversarial robustness of HDC by developing a systematic approach to test and enhance the robustness of HDC against adversarial attacks with two main components: (1) TestHD, which is a highly-automated testing tool that can generate high-quality adversarial data for a given HDC model; and (2) GuardHD, which utilizes the adversarial data generated by TestHD to enhance the adversarial robustness of HDC models. The core idea of TestHD is built on top of fuzz testing method. We customize the fuzzing approach by proposing a similarity-based coverage metric to guide TestHD to continuously mutate original inputs to generate new inputs that can trigger incorrect behaviors of HDC model. Thanks to the use of differential testing, TestHD does not require knowing the labels of the samples beforehand. For enhancing the adversarial robustness, we design, implement, and evaluate GuardHD to defend HDC models against adversarial data. The core idea of GuardHD is an adversarial detector which can be trained by TestHD-generated adversarial samples. During inference, once an adversarial sample is detected, GuardHD will override the prediction result with an "invalid" signal. We evaluate the proposed methods on 4 datasets and 5 adversarial attack scenarios with 6 adversarial generation strategies and 2 defense mechanisms, and compare the performance correspondingly. GuardHD is able to differentiate between benign and adversarial inputs with over 90% accuracy, which is up to 55% higher than adversarial training-based baselines. To the best of our knowledge, this paper presents the first comprehensive effort in systematically testing and enhancing the robustness against adversarial data of this emerging brain-inspired computational model.**

***Index Terms*—hyperdimensional computing, differential fuzz testing, adversarial attack, robust computing**

I. Introduction

Hyperdimensional computing (HDC) is an emerging computing scheme based on the working mechanism of brain that computes with deep and abstract patterns of neural activity instead of actual numbers. Compared with machine learning (ML) algorithms such as DNN, HDC is more memory-centric, granting it advantages such as relatively smaller model size, less computation cost, and one-shot learning capabilities, making it a promising candidate in low-cost computing platforms [1]. Recently, HDC has demonstrated promising capability on various applications such as language classification [2], vision sensing [3], brain computer interfaces [4], gesture recognition [5], and DNA pattern matching [6]. However, despite the growing popularity of HDC, the discussions on the reliability and robustness of HDC models are relatively limited.

The traditional approach to testing ML systems is to curate a specific set of data with corresponding labels and input them into the system to assess the accuracy. However, as the ML systems are scaling significantly and the input space is becoming more sophisticated, such approach is hardly feasible and scalable anymore. On the other side, researchers have found that by adding even invisible perturbations onto original inputs to create "adversarial attacks", ML systems can be "fooled" and produce wrong predictions [7], [8]. Just like DNNs, HDC can also be vulnerable to small perturbations on inputs, as shown in Fig. I. This brings a dire need of frameworks to automatically generate high quality adversarial samples to attack and test the HDC model as well as detection and defense mechanisms that can leverage the generated samples to enhance the robustness against attacks. However, attempting to developing such a framework for HDC models faces challenges from both sides on testing and defense.



(a) (b) (c)

Fig. 1. An example of adversarial image of HDC by mutating some pixels in the image:(a) the original image as "8"; (b) the pixels mutated; (c) the mutated image wrongly predicted as "3".

Challenges on Adversarial Testing: Unlike traditional ML systems such as DNNs with a well-defined mathematical formulation and relatively fixed architecture (specific layer types and network structures), HDC is not differentiable and less application-agnostic. The encoding of HDC is largely unique for each application and relies on random indexing to project data onto vectors in a hyperdimensional space [2], adding difficulty to efficiently acquire adequate information to guide the adversarial generation process. As a result, gradient-related

Corresponding Author: Xun Jiao.

D. Ma and X. Jiao are with the Department of Electrical and Computer Engineering of Villanova University, Villanova, PA 19085. (e-mail: {dma2, xun.jiao}@villanova.edu).

T. S. Rosing is with the Department of Computer Science and Engineering, University of California, San Diego, La Jolla, CA 9209 (e-mail: tajana@ucsd.edu).

generation techniques used in DNNs cannot be directly applied here since they rely on a set of well-defined mathematical optimization problems as well [9]. Furthermore, the standard approach to testing ML systems is to gather and label as much real-world test data as possible [10], [11]. Google even used simulation to generate synthetic data [12]. However, such effort is not only largely short of scalability but also unguided as it does not consider the internal structure of ML systems, making it unable to cover more than a tiny fraction of all possible corner cases.

Challenges on Adversarial Detection: Existing defense mechanisms for ML models are also not applicable to HDC models. For example, the adversarial learning approaches [13] used in robust ML training will degrade the prediction accuracy of HDC models [14]. Moreover, adversarial training based algorithms require extensive off-line or even "posthumous" retraining and fine-tuning on specific models, resulting in less flexibility and difficult integration into the existing HDC systems particularly with real time requirements. On the other hand, the existing works are often parochial, i.e., specific to one or few attack scenarios such as gray-box attacks. Such detection and defense methods may also suffer from transferability deficiencies in realistic implementations as the source of the attacks can come from unknown or rather diverse sources.

To address the above-mentioned challenges, we present an effort to enhance the robustness of HDC model against adversarial attacks, which consists of two major components: 1). **TestHD**, built on top of our previous work **HDTest** [15], for highly-automated adversarial generation based on differential fuzz testing. Fuzz testing is a software testing technique that strategically mutates inputs with the goal of generating faults or exceptions automatically [16], [17]. **TestHD** can be used in both black-box (unguided) and gray-box (coverage-guided) scenarios, making it a highly scalable testing solution. 2). **GuardHD**, a "sub-HDC model" dedicated to verify if the input samples are benign or adversarial and can override potential erroneous predictions caused by adversarials with invalid signals. As **GuardHD** is also an HDC classifier, it can be easily appended to the existing HDC model and can be executed in parallel with the the original model inference while having minimal impact compared with adversarial training based defense algorithms.

Our main contributions are as follows:

- We present a systematic effort to test and enhance the HDC robustness against adversarial attacks. Based on differential fuzz testing, **TestHD** iteratively mutates inputs to generate new inputs that can trigger incorrect behaviors of HDC models without the necessity of knowing the label. **GuardHD**, on the other hand, formulates an HDC detector model which provides detection and defense methodologies.
- **TestHD** develops various mutation strategies to generate the inputs. **TestHD** leverages unique property of HDC and introduce the guided fuzz testing based on the similarity to improve fuzzing efficiency.
- **GuardHD** leverages the concept of "sub-model" into HDC and propose **GuardHD**. **GuardHD** is an HDC model dedicated to detect if an input sample is adversarial. If so, it will override the original HDC output with an "invalid" signal, to indicate an attempt to attack is detected.
- We use four datasets: MNIST, Medical MNIST, MASK and FACE to evaluate **TestHD** and **GuardHD**. We compare the efficiency and quality of the generated samples under different scenarios and strategies. Further, trained using the adversarial samples generated by **TestHD**, **GuardHD** can identify and defend the adversarial samples with over 90% accuracy on average, which is up to 55% higher than the adversarial training-based defense methods [14], [18] across diverse scenarios. We also perform design space exploration on **GuardHD** to further evaluate and analyze its performance under different HDC configurations.

In Sec. II, we present the related works as well as highlight the main novelty of this paper. In Sec. III, we provide the necessary background and preliminaries on HDC. Sec. IV discuss **TestHD** and how it can generate quality adversarial samples given an HDC model under test while Sec. V talks about how to defend the adversarial attacks using **GuardHD**. We present the experimental results of evaluating **TestHD** and **GuardHD** at Sec. VI and Sec. VII respectively. We also discuss the potential directions for future work at Sec. VIII and conclude the paper at Sec. IX.

II. Related Work

A. Hyperdimensional Computing

Since Kanerva's first introduction of HDC for learning tasks [19], HDC has been applied to various emerging application domains [20]. In IoT, HDC is deployed in edge devices for bio-signal (such as EEG, ECG and EMG) processing to accurately detect seizures [21], or recognize hand gestures recognition with 97.8% accuracy on average which surpasses support vector machine by 8.1% [22]. HDC also shows superior accuracy to neural networks in speech recognition as well as smaller model size and memory footprint [23]. Moreover, biological sequence matching applications such as DNA sequencing also experiences 2X to 4X speed-up when using HDC as well as 5% accuracy increase [24]. HDC has also been applied to radar systems for energy efficient and faster classification of indoor human activities with comparable accuracy [25]. For optimization of HDC processing, *HDC-IM* [26] proposed in-memory computing techniques for HDC scenarios based on Resistive RAM. There are also optimizations on HDC targeted at different computing platforms such as FPGA [27] and 3D IC [28].

As HDC is increasingly applied into security-critical domains recently, more related literature on HDC security and privacy emerges. By applying genetic algorithms or software fuzzing, adversarial images are generated for HDC that can trigger wrong predictions [14]. In addition, HDC-based voice recognition systems is also found vulnerable to adversarial attacks with small perturbations added in the audio sample [29]. System-wise, HDC is also subject to other security concerns such as privacy leak [30] and IP stealing [31]. These studies highlight the significant necessity of defense mechanisms to offer protection against such attacks and enhance the model robustness under input perturbations.

B. Adversarial Attack and Defense in DNNs

Recently, adversarial deep learning have demonstrated that state-of-the-art DNN models can be fooled by crafted synthetic images with minimal perturbations added to an existing image. Goodfellow *et al.* proposed a fast gradient sign method of generating adversarial examples with required gradient computed efficiently using backpropagation [9]. Nguyen *et al.* calculated the gradient of the posterior probability for a specific class (e.g., softmax output) with respect to the input image using backpropagation, and then used gradient to increase a chosen unit's activation to obtain adversarial images [32]. As to defense, enhanced input processing such as denoising algorithms can sanitize the input, i.e., mitigate the perturbations maliciously added [33], [34]. Another line of defense methods leverage the natural classification capability of machine learning models to develop a classifier, as a "sub model" or "adversarial detector" dedicated to identify attacks [35], [36]. Techniques such as dimension reduction can be implemented together with the adversarial detector as a method to defend and eliminate noise that pertains to the attack [37]. The attack samples can also be injected into the training data during adversarial training to increase the model robustness [38], [39].

C. Main Novelty

We highlight the main novelty of this work from two perspectives: adversarial sample generation by **TestHD** and adversarial defense by **GuardHD**. (1) As to adversarial sample generation in HDC, while there is gradient-less method in generating adversarial samples such as using genetic programming [40], our work is the first to enabling adversarial generation without the necessity of labels thanks to the use of differential testing, which expands the scalability and flexibility of adversarial sample generation. Further, while fuzzing methods have been widely applied in traditional software testing, this is the first time fuzzing method has been applied to HDC and we customize the fuzzing flow by developing the novel HV similarity coverage and use it to guide fuzzing process. (2) As to adversarial attack defense, this is the first time an adversarial detection method has been developed for HDC models, and we have, according to the specific characteristics of HDC, customized the adversarial detection method by developing two representative HVs where each of them represent either benign sample or adversarial sample. Our work also considers the performance of adversarial detection under different attack scenarios, and provides design space exploration by varying HV dimensions and data-types. Last but not least, we also compare **GuardHD** with the state-of-the-art adversarial defense methods [14], [15] in the HDC community.

III. HDC Background

This section provides the necessary backgrounds and preliminaries on understanding how HDC model works as an algorithm in learning tasks.

A. Notions and Preliminaries

Hypervectors Hypervectors (HV) are the "building blocks" of HDC models. They are high-dimensional holographic vectors with i.i.d. elements (often numbers) [19]. An HV with d dimensions can be denoted by Eq. 1, where h_i is the i-th element. Within an HDC model, the dimension of HVs is usually consistent.

$$\vec{H} = \langle h_1, h_2, \ldots, h_d \rangle \tag{1}$$

Operations As HVs are vectors, we can perform different vector operations using HVs to aggregate information. Addition ($+$), multiplication ($*$) and permutation (ρ) are the three common operations that are usually used in HDC, as shown in Eq. 2. Addition and multiplication take two input HVs and perform **element-wise** add or multiply operations. Permutation takes one HV and perform **cyclic shift** by certain dimensions. Note that the dimension of HV is not modified during all these three operations.

$$\begin{aligned} &\vec{H_p} + \vec{H_q} = \langle h_{p1} + h_{q1}, h_{p2} + h_{q2}, \ldots, h_{pd} + h_{qd} \rangle \\ &\vec{H_p} * \vec{H_q} = \langle h_{p1} * h_{q1}, h_{p2} * h_{q2}, \ldots, h_{pd} * h_{qd} \rangle \\ &\rho_1(\vec{H}) = \langle h_d, h_1, h_2, \ldots, h_{d-1} \rangle \end{aligned} \tag{2}$$

The operations also have their corresponding "realistic" meanings [19]. Addition is used to aggregate 'parallel" information, i.e., information from the same modality. Multiplication is used to combine information from different sources or modalities to produce another layer of information. Permutation is used to reflect spatial or temporal changes in the information which often occur during a sequence or series of data.

Similarity Measurement In HDC, every HV possesses a certain information. The similarity ζ between the two HVs indicates the affinity between the information they correspondingly possess. Different algorithms can be used to calculate the similarity, such as the Euclidean ($L2$) distance, the Hamming distance (for binary HVs), and cosine similarity (which we use in this paper as noted in Eq. 3). Higher similarity means higher affinity, i.e., more information in common between the two HVs.

$$\zeta(\vec{H_p}, \vec{H_q}) = \frac{\sum_{i=1}^{d} h_{pi} \cdot h_{qi}}{\sqrt{\sum_{i=1}^{d} {h_{pi}}^2} \cdot \sqrt{\sum_{i=1}^{d} {h_{qi}}^2}} \tag{3}$$

B. Developing an HDC Model

As Fig. 2, there are 3 key phases in developing an HDC model for classification tasks: **Encoding**, **Training**, and **Inference**.

Encoding Encoding is the fundamental process in developing an HDC model, which is to project real-world features into their high-dimensional space representations – the HV. Encoding features a combination of HV operations over the item memory, whereas such combination is developed and specified according to the application. **Item memory** stores randomly generated item HVs, each representing a unique feature value according to the realistic properties of the feature.

Encoding can be described by Eq. 4. Assume each sample has m input features : $\vec{F} = \langle f_1, f_2, \ldots, f_m \rangle$, thus there are m item memories $\mathcal{R}$ corresponding to each feature $\mathcal{R} = \{\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_m\}$. Assume the application-specific combination is Γ, therefore, for each feature in the sample, we can index its corresponding item HV from the item memory. Using Γ, those item HVs are then encoded into the sample HVs $\vec{H}$ which will subsequently represent the sample in the model development of HDC (training and inference).

$$\vec{H} = \Gamma(\mathcal{R}, \vec{F}) = \Gamma(\mathcal{R}_1[f_1], \mathcal{R}_2[f_2], \ldots, \mathcal{R}_m[f_m]) \quad (4)$$

Training Training is to aggregate information of samples from the same class. Training adds up the encoded hypervectors sharing the same label into class HVs in a dedicated associative memory $\mathcal{A}$. Considering a classification task with k classes, as every class HV in the associative memory stands for a class, thus there are k class HVs in the associative memory. The associative memory is initialized to zeros and for each sample in the training set. Each sample HV $\vec{H}^l$ with label l is then added into $\mathcal{A}$ iteratively. This process of training can be denoted as Eq. 5.

$$\begin{aligned} \mathcal{A} &= \{\vec{A}^1, \vec{A}^2, \ldots, \vec{A}^k\} \\ &= \{\sum \vec{H}^1, \sum \vec{H}^2, \ldots, \sum \vec{H}^k\} \end{aligned} \quad (5)$$

Inference Inference is to predict the class of unseen input samples. In HDC, inference is based on checking the similarity between the HV representing the unseen sample (i.e., the query HV, $\vec{H}_q$) and every class HV in the associative memory. $\vec{H}_q$ is encoded using the same combination of HD operations Γ as other samples. The class of the highest similarity with the query HV is selected as the predicted label l_p for this unseen sample.

$$l_p = \text{argmax}(\{\zeta(\vec{H}_q, \vec{A}^1), \zeta(\vec{H}_q, \vec{A}^2), \ldots, \zeta(\vec{H}_q, \vec{A}^k)\}) \quad (6)$$

IV. TestHD: Adversarial Generation

In this section, we introduce how we target at an HDC classifier and generate the adversarial attack images under different scenarios such as black-box and gray-box scenarios, where the images will be used when evaluating **TestHD**.

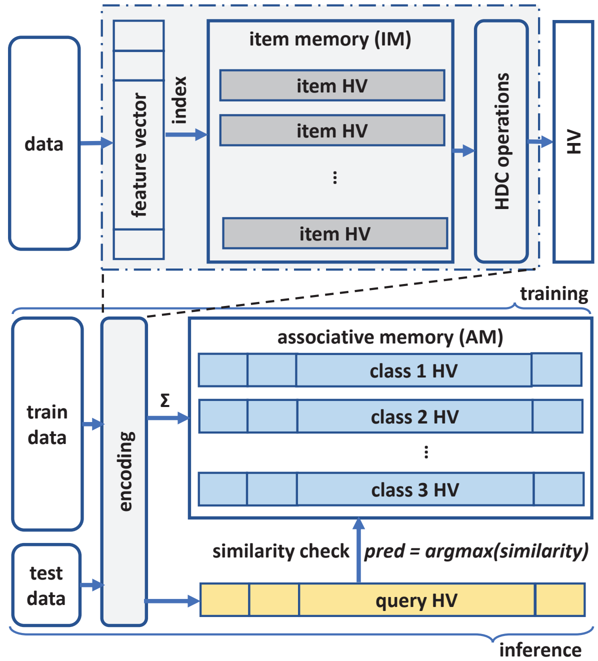


Fig. 2. Overview of Hyperdimensional Computing on Classification tasks: Encoding, Training and Inference.

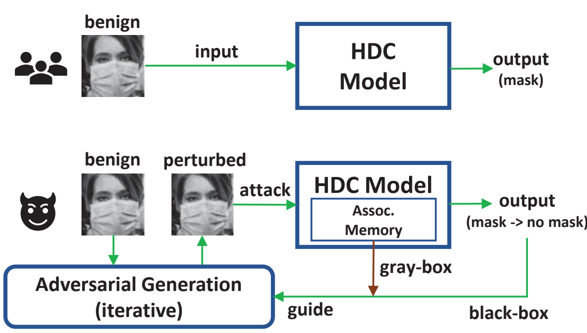


Fig. 3. Different behavior between a regular user and an attacker.

A. Threat Model

Assume we have an HDC model deployed for mask detection as an example shown in Fig. 3, a regular user inputs an head photo to the HDC model and obtains output labels, i.e., whether the person in the head photo is wearing a mask or not. An attacker however, aims to generate perturbed adversarial photos that can successfully attack, or fool the model to output a wrong label for prediction so that person not wearing a mask is predicted as wearing a mask, or vice versa. To minimize the perturbation visually, the generated photos are required to resemble the original benign photo as much as possible.

We assume the attacker, same as a regular user, can input images to the target HDC model and obtain corresponding output labels. We assume the target HDC model under attack has the associative memory $\mathcal{A}$, and a benign image of size $W \times H$ as an input sample to the classifier is $\mathcal{X} \in X^{W,H}$. Using the encoding process from Sec. III, we can obtain the encoded sample HV for this image, $\vec{H_x} = \Gamma(\mathcal{R}, \mathcal{X})$. According to Eq. 6, the HDC model will make prediction l_x based on the highest similarity between the sample HV $\vec{H_x}$ and the class HVs inside the associative memory $\mathcal{A}$.

The attacker tries to modify the benign image $\mathcal{X}$ by adding perturbation $P^{W,H}$, with the objective to generate adversarial image $\mathcal{X} + \mathcal{P}^{W,H}$ so that, the sample HV obtained from the same encoding $H_{\mathcal{X}+\mathcal{P}^{W,H}} = \Gamma(\mathcal{R}, \mathcal{X} + \mathcal{P}^{W,H})$ can make the HDC model to produce a different prediction, i.e., $l_{\mathcal{X}+\mathcal{P}^{W,H}} \neq l_{\mathcal{X}}$. The attacker also wants to minimize the difference between the original image and the generated adversarial image according to the similarity metrics $\mathcal{L}$. We can formulate this as an optimization problem as Eq. 7.

$$\begin{aligned} \min \quad & \mathcal{L}(\mathcal{X} + \mathcal{P}^{W,H}, \mathcal{X}) \\ s.t. \quad & l_{\mathcal{X}+\mathcal{P}^{W,H}} \neq l_{\mathcal{X}} \end{aligned} \tag{7}$$

Note that the attacker can use different adversarial generation strategies and the scenarios can be diverse as well (black-box or gray-box). In realistic implementations, the performance of detection and defense method can drastically vary. Therefore, we provide a further discussion on the attack scenarios and generation strategies in the following section. A comprehensive evaluation on **TestHD** performance across the scenarios and strategies is presented in Sec. VI.

B. Attack Scenario

In neural networks, algorithms can leverage different parameters such as calculating the gradient with the cost functions (e.g. cross-entropy) to guide the adversarial image generation to achieve the optimization goals [41], [42]. However in HDC, due to the fundamental difference on the model construction, i.e., prediction of HDC model is based on hypervector similarity, and parameters used in neural networks like gradient and cross-entropy are typically not available.

We assume two possible attack scenarios in this paper: black-box and gray-box. Under the (hard-label) black-box scenario, attackers are not able to gain access to the internal establishments of the HDC model such as item memories and associative memories, i.e., attackers can only input images to the HDC model and observe the prediction labels. Therefore, generation of adversarial images uses different pre-defined perturbation mechanisms which are directly aggregated with the original image to produce the modified image and check if the predicted label becomes different. On the other hand, under the gray-box scenario, attacker is able to obtain the similarity metrics such as Hamming distance or cosine similarity between the query HV of the image and the class HV in the associative memory. Therefore, these metrics can be leveraged to quantitatively guide the adversarial image generation in HDC models to generate higher quality attack images compared with those from the black-box scenario.

C. Adversarial Generation

The overview of **TestHD** is illustrated in Fig. 4. **TestHD** takes the original input image t without necessarily knowing the label of it. **TestHD** then applies mutation algorithms (strategies) on the original input t to generate new input t'. Both the generated input and the original input are then sent to the HDC classifier for prediction. We then check if the two predicted labels are different, and if yes, this indicates a successful generation of an adversarial input. Otherwise, **TestHD** will continue repeating the fuzzing process.

The mutation algorithm of **TestHD** is shown in Alg. 1. Its objective is to generate adversarial images by applying mutation strategies to change, or add perturbation on the image. For each input in the unlabelled input image dataset, first, **TestHD** uses the HDC model to get the predicted label of the original input image (Line 3) as a reference label. Then, **TestHD** applies mutation strategies on the original image to generate different mutated images as seeds (Line 5). We use 20 for the number of seeds and 5 for the number of survivors for each iteration, and user can also specify the desired amount of seeds generated and survived. Again, **TestHD** feeds the seeds into HDC and obtains the corresponding label of the seeds as query labels (Line 6). By comparing the query labels with reference labels, **TestHD** is able to know if there are any discrepancies which indicate successful generation of an adversarial image (Line 7–10). The adversarial images are added to the set of adversarial samples and **TestHD** proceeds to the next image. If all the seeds are still predicted the same as the original input image, **TestHD** will select the survival seeds and repeat the process, until a successful adversarial image is generated, or the maximum allowed $iter\_times$ is reached (Line 4). Users can further customize $iter\_times$ as the budget for evaluating robustness of different HDC models can vary based on the actual use-case and scenario.

However, how the seeds are selected to survive into the next iteration is different between gray-box and black-box scenarios. For black-box scenarios, the survival seeds are randomly sampled from all the seeds (Line 13). For gray-box scenarios, **TestHD** uses guided mutation for seeds update based on the similarity that only the top fittest seeds can survive. The coverage ω of a seed is defined as Eq. 8, where $\mathcal{A}[y]$ is the class y's HV of in the associative memory and $\vec{H}_{y'}$ is the query HV of the seed, encoded by the HDC model. In fuzz testing for other machine learning models such as DNNs, coverage is usually defined by neuron coverage and higher coverage usually indicates higher possibility to trigger exceptional behaviors of the model [41], [42]. Although HDC does not explicitly have neurons like neural networks, we similarly define the coverage based on the similarity metrics using the associative memory. Here in HDC, higher coverage means lower similarity between the HV of the seed and the original input image's HV, indicating higher possibility to generate an adversarial image, i.e., to trigger exceptional behavior of the model.

$$\omega = 1 - \zeta(\mathcal{A}[y], \vec{H}_{y'}) \tag{8}$$

To ensure the added perturbations are within the desired range, we need quality metrics to evaluate the samples gener-

This article has been accepted for publication in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. This is the author's version which has not been fully edit
content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2023.3263120

TABLE I
TESTHD STRATEGIES OF ADVERSARIAL GENERATION

strategy	description
gauss	Add Gaussian noise onto image
rand	Add random noise onto image
stripe	Randomly set certain columns to create stripes
blur	Apply Gaussian blur onto image
erosion	Apply erosion onto image
dilation	Apply dilation onto image

ated. One of the most applied metric is the distance between the generated samples and the original samples, such as $L2$ or $L1$ distances [40]–[42]. User can also implement their desired metrics as a method and incorporate in **TestHD**. The quality metrics can work with coverage in a synergy to control the adversarial generation, for example, **TestHD** enables users to set a hard threshold of such metrics so that when the distances are beyond it, the generated image is regarded as unacceptable and then discarded. On the other hand, the quality metrics can also be incorporated in calculating the coverage as a regularization during the fuzzing process.

We adopt 6 typical adversarial generation strategies for HDC models as listed in Table I, including adding various types of noises and applying morphological processing [14], [18].

Algorithm 1 TestHD Generation Algorithm

Input inputs: unlabeled input images for testing.
HDC: the HDC model under test.
mutate: mutation strategies listed in Table I.
Output S: (set of) adversarial (input) images.

```
S = []
for t in inputs do
    y = HDC(t)
    while max allowed iterations not reached do
        t' = mutate(t)
        y' = HDC(t')
        for y' in y' do
            if y' != y then
                S.append(y')
                break
            else
                if black-box scenario then
                    seeds = seeds.select_random()
                else if gray-box scenario then
                    seeds.calc_coverage(HDC.am)
                    seeds = seeds.select_fittest()
                end if
            end if
        end for
    end while
end for
```

V. GUARDHD: ADVERSARIAL DEFENSE

In this section, we introduce how **GuardHD** is developed to defend the attacks from the adversarial samples generated.

A. Defense Mechanism

We formulate the adversarial attack defense task as a binary classification problem, i.e., classifying the input sample to be adversarial or not. In **GuardHD**, we augment our HDC model by adding a “sub model” as such detector to classify the input sample concurrently with the inference of the existing HDC model. Specifically, we append an additional associative memory – “**GuardHD** detection memory” into the existing system. This memory accommodates two class hypervectors, each representing the class of adversarial and benign input sample, respectively. We configure the memory using the same parameters such as HV dimension and data type with the existing HDC model for ease implementation and integration.

We illustrate the defense mechanism of **GuardHD** in Fig. 5 with two cases: benign and adversarial input samples. In both cases, the input is encoded using the same encoding scheme and item memory as described in Sec. III to obtain the query HV of the input. The existing HDC model checks the similarity between the query HV and each of the class HVs in the associative memory to output the prediction by the class with highest similarity.

However, **GuardHD** has different behaviors when the input is identified as benign and adversarial. When existing HDC model is making an inference, **GuardHD** obtains a copy of the query HV and checking the similarity between the query HV and each of the two HVs inside the detection memory. If the query HV is more similar to the benign HV, the input sample is detected as benign, so the prediction from the existing HDC model can be normally released from the multiplexer as the final prediction result. On the other hand if the query HV is more similar to the adversarial HV, the input sample is instead detected as adversarial. **GuardHD** then controls the multiplexer so that an “invalid” signal will be released instead to override the prediction as the final output result. This indicates that a potential adversarial input sample is identified so that the corresponding prediction will not be passed to the system.

B. Training of *GuardHD*

Training of **GuardHD** resembles the training of a typical HDC classifier model. In general HDC model the number of class HVs in the associative memory matches the number of classes in the learning task. In contrast, **GuardHD** is always a binary classifier, which means it requires only two HVs (representing benign or adversarial samples) in the added detection memory. Thus, the memory overhead introduced by **GuardHD** is always fixed. The training samples are only labelled as benign or adversarial, regardless of what their original label is. The samples inside the original dataset can be directly used as benign samples. The adversarial samples are generated based on the benign samples as described above, with the generation strategies mentioned in **TestHD**.

This article has been accepted for publication in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. This is the author's version which has not been fully edi
content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2023.3263120

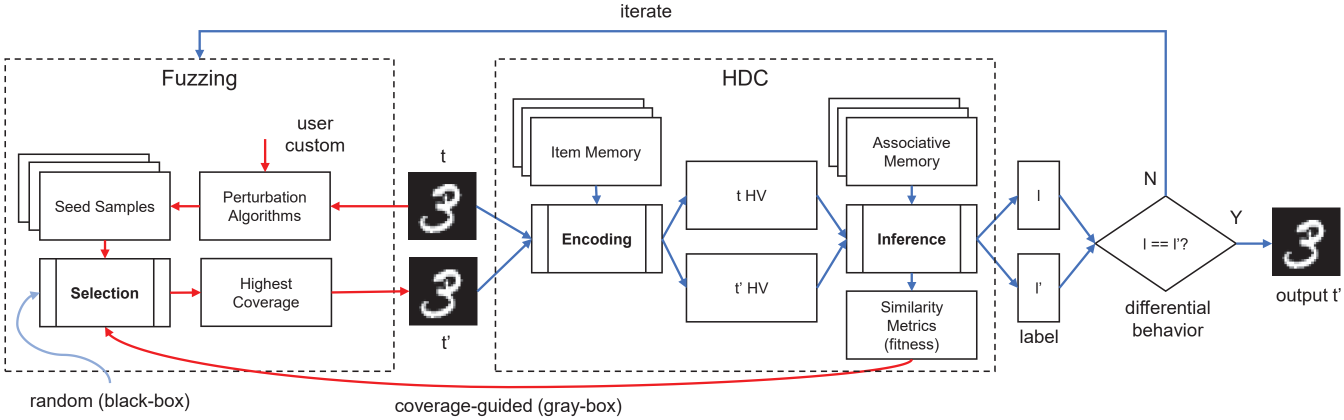


Fig. 4. Overview of **TestHD**

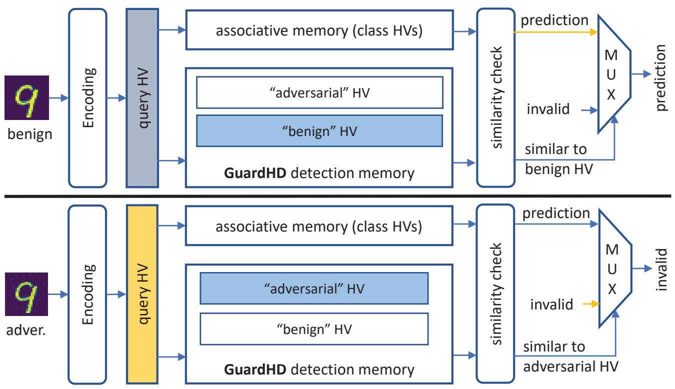


Fig. 5. Workflow of **GuardHD** framework with benign input and adversarial input samples.

VI. Evaluation of TestHD

A. Experimental Setup

Dataset: Most of the existing works on HDC focus on two datasets of MNIST and FACE for vision tasks as HDC is usually recognized as an ultra light solution for edge computing systems [43]–[46]. We expand to four datasets by adding **Medical MNIST** and **MASK** as detailed below to evaluate **TestHD** and **GuardHD**.

We show example adversarial samples generated from black-box and gray-box scenarios for all the datasets with corresponding original samples in Fig. 6 and Fig. 7. The images are post-processed (resized) for presentation.

- **MNIST:** the handwritten digit classification datasets with 10 classes from number 0 to 9 [47].
- **Medical MNIST:** the medical imaging dataset of 6 classes: abdomen CT, breast MRI, CXR, chest CT, hand CT, and head CT [48].
- **MASK:** the head photo dataset of 2 classes where the person in the photo is wearing a face mask or not [49].
- **FACE:** the human face recognition dataset to classify if the image contains a human face or not, where the images are scraped from CIFAR-10 [50] and the celebrity faces (CelebFaces) Dataset [51].

Quality Metrics: Since the images of the datasets are in different size, we normalize $L1$ and $L2$ distance with the number of pixels in each image. We also scales dataset with floating point range $(0-1)$ into integers $(0-255)$ to make the distance metrics consistent.

B. Quality of Adversarial Generation

TABLE II
Adversarial Sample Quality: Black-box vs. Gray-box

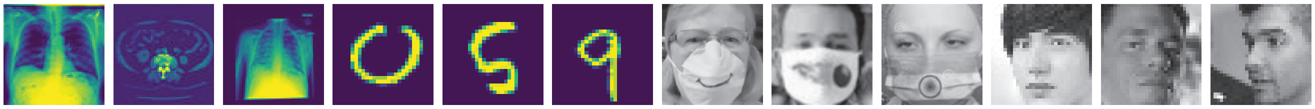
		MNIST	Medical MNIST	FACE	MASK
black-box	L1	10.127	6.741	9.400	4.670
	L2	1.121	0.256	0.574	0.074
gray-box	L1	9.851	6.506	9.237	4.731
	L2	1.093	0.246	0.570	0.076

TABLE III
Adversarial Sample Quality: Strategies

		MNIST	Medical MNIST	FACE	MASK
stripe	L1	8.435	2.979	4.422	1.133
	L2	1.640	0.386	0.820	0.106
random	L1	2.456	3.936	5.042	4.500
	L2	0.127	0.076	0.190	0.042
gaussian	L1	4.685	10.259	9.406	8.843
	L2	0.252	0.207	0.369	0.087
erosion	L1	16.443	6.677	13.377	5.977
	L2	1.884	0.281	0.765	0.097
dilation	L1	16.964	11.108	14.402	5.437
	L2	1.910	0.397	0.855	0.088
blur	L1	10.951	4.783	9.261	2.314
	L2	0.831	0.160	0.435	0.032

We present the average quality metrics of generated adversarial samples under different scenarios (Table II) and using different generation strategies (Table III) where we can make several observations. For example, we notice that the quality of the adversarial samples generated under gray-box scenario is slightly higher, with 2% – 4% lower distance to the original

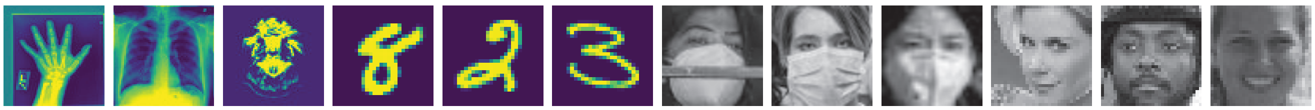
This article has been accepted for publication in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2023.3263120



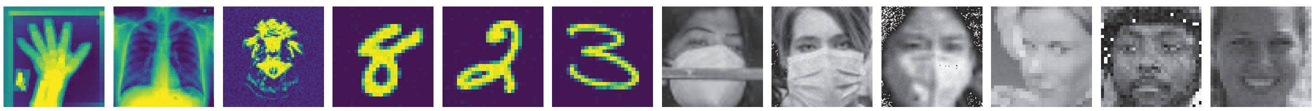
(a) original samples

(b) generated (adversarial) samples

Fig. 6. Adversarial samples generated under the black-box scenario and their corresponding original samples.



(a) original samples



(b) generated (adversarial) samples

Fig. 7. Adversarial samples generated under the gray-box scenario and their corresponding original samples.

image on average for three out of the four datasets: MNIST, Medical MNIST and FACE. We also observe that the quality metrics can drastically vary between different strategies of generation. For example, samples generated from gaussian and random strategies are usually with lower distance from the original image and samples generated from morphological processing such as erosion and dilation are having much larger distances. This is expected due to the natures of those generation strategies. The noise added to the pixels in the original image under the random and gaussian strategies are following corresponding distributions that only a few pixels will be significantly perturbed.

However, since the erosion and dilation techniques are applied to the entire sample image, the distance can be significantly higher, as all the pixels are changed according to the morphological kernels. The metrics are also related to the size of the image, e.g., adding a stripe of noise to a 28×28 image (MNIST) dataset will be quite a noticeable perturbation since $\frac{1}{28} \approx 3.5\%$ of the total pixels are mutated, while for an 128×128 image (FACE), only $\frac{1}{128} < 1\%$ of the total pixels in the image are changed. This also indicates that traditional metrics such as $L1$ and $L2$ may not always be appropriate in describing the quality of generated samples.

C. Efficiency of Adversarial Generation

We also present the average generation time of each adversarial sample under different scenarios and strategies in Table IV. We can observe that for gray-box scenarios, the generation time is considerably higher, since the guided seed selection requires HDC model inference on each seed sample and comparing the similarity metrics. As to strategies, random and gaussian require much more time to complete since they require more iterations of seed selections to generate an adversarial sample. For morphological processing, most of the adversarial samples are generated with one iteration since any additional iteration is likely to excessively perturb of the original sample. In addition, larger images also requires longer time of generation as both applying the perturbations and using HDC model to make inference will take additional time to accomplish.

TABLE IV
GENERATION TIME (SEC) PER ADVERSARIAL SAMPLE UNDER DIFFERENT SCENARIOS AND STRATEGIES

	MNIST		Medical MNIST	
	black-box	gray-box	black-box	gray-box
stripe	0.046	0.119	0.238	0.575
random	0.498	1.520	2.386	7.646
gaussian	0.427	1.281	1.341	3.937
erosion	0.076	0.129	0.256	0.439
dilation	0.102	0.164	0.521	0.856
blur	0.285	0.469	1.133	2.148
	FACE		MASK	
	black-box	gray-box	black-box	gray-box
stripe	0.079	0.182	0.207	0.843
random	0.887	2.712	2.429	10.007
gaussian	0.588	1.816	1.822	7.428
erosion	0.414	0.655	0.575	1.456
dilation	0.146	0.230	0.460	1.186
blur	0.518	0.813	1.000	3.296

VII. EVALUATION OF **GUARDHD**

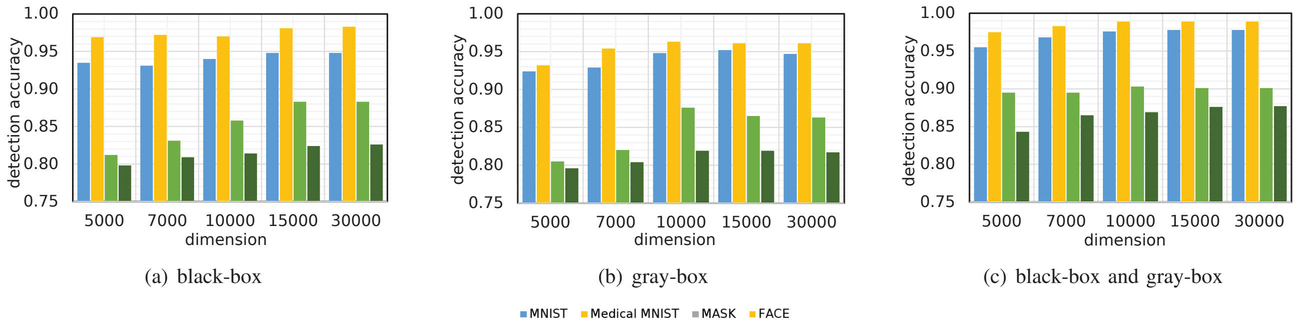
A. Detection Accuracy of ***GuardHD***

GuardHD performance is evaluated by detection accuracy, i.e., the classification accuracy of predicting the input to be

This article has been accepted for publication in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2023.3263120

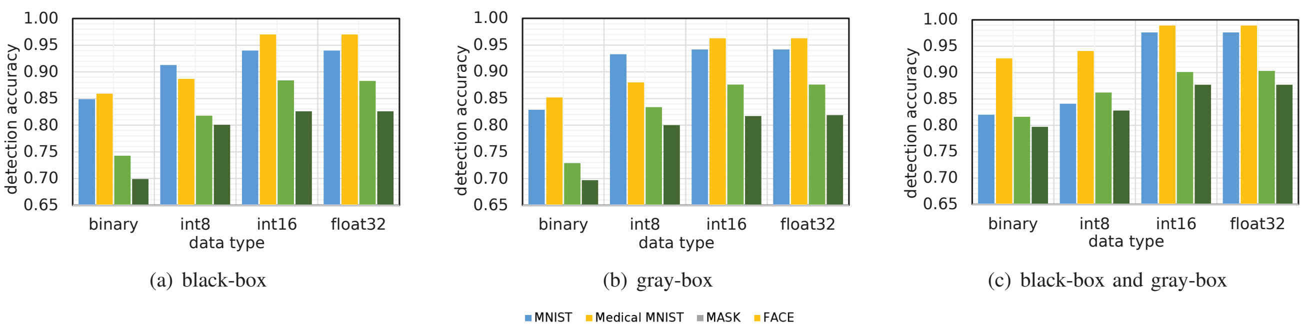
TABLE V
EVALUATION CONFIGURATION AND DETECTION ACCURACY, RECALL AND PRECISION OF **GUARDHD**

train/test set	MNIST detection accuracy	MNIST recall	MNIST precision	Medical MNIST detection accuracy	Medical MNIST recall	Medical MNIST precision
black-box/black-box	0.948	0.933	0.951	0.97	0.949	0.991
gray-box/gray-box	0.941	0.941	0.944	0.963	0.954	0.978
black-box+gray-box/black-box+gray-box	0.976	0.906	0.97	0.989	0.982	0.994
train/test set	**MASK detection accuracy**	**MASK recall**	**MASK precision**	**FACE detection accuracy**	**FACE recall**	**FACE precision**
black-box/black-box	0.883	0.802	0.885	0.819	0.886	0.737
gray-box/gray-box	0.876	0.877	0.823	0.826	0.797	0.797
black-box+gray-box/black-box+gray-box	0.903	0.972	0.873	0.877	0.919	0.857



(a) black-box (b) gray-box (c) black-box and gray-box

Fig. 8. Impact of dimension on the adversarial attack defense performance of **GuardHD**.



(a) black-box (b) gray-box (c) black-box and gray-box

Fig. 9. Impact of data type on the adversarial attack defense performance of **GuardHD**.

adversarial or benign. According to Table V, first we examine the performance of **GuardHD** under individual scenario, i.e., train and evaluate on **either** black-box **or** gray-box samples. We then evaluate **GuardHD** with all samples together, i.e., train and evaluate on **both** black-box **and** gray-box samples.

In addition to detection accuracy, we also present two other metrics: recall and precision of **GuardHD** in Table V as **GuardHD** is essentially a binary classification model. As detection accuracy is **GuardHD**'s accuracy on identifying adversarial samples, recall is to measure the successfully identified attacks over all attacks and precision is to assess the rate of correctly identified attacks over all identified attacks. Those additional metrics help with monitoring the false positives in detection adversarial attacks.

For all of the four datasets, **GuardHD** is able to achieve detection accuracy higher than 80%. This indicates that the 100% attack success rate of the attack set is now reduced to less than 20% using **GuardHD**. Compared with adversarial training methods, this can achieve up to 55% additional attack success rate reduction [14], [15]. The highest detection accuracy is for the Medical MNIST under the scenario of having both the black-box and gray-box generated samples. Except for the FACE dataset, we can observe that gray-box generated adversarial samples are usually more difficult to detect. However, by using the adversarial attack samples generated from both scenarios together to train **GuardHD**, the detection accuracy of **GuardHD** can increase by 2% – 5% across the four datasets.

The recall scores of all the scenarios are also above 80% for most scenarios, indicating **GuardHD** has a high capability of successfully defending adversarial attacks with just a small amount of false negatives. All the precision scores are also higher than 80% except for FACE dataset, which shows that **GuardHD** has just a small impact on the performance of the original HDC model.

B. Design Space Exploration on GuardHD

As **GuardHD** is also an HDC model, we also perform its design space exploration by evaluating the impact of two **GuardHD** hyper-parameters related to HDC: the dimension and the data type of HV. We sweep the dimensions from 5,000 to 30,000 and evaluate the data types from binary to 32-bit integer as well as the single precision 32-bit float. Such design space exploration can help identify optimal configurations which is necessary especially when **GuardHD** is equipped with HDC models that are deployed on resource limited systems such as edge computing platforms or embedded architectures.

As to HV dimension, from Fig. 8, we could observe a general trend that when dimension decreases from 30,000 to 5,000, the detection accuracy suffers from degradation across all the datasets. This is because an HV with a lower dimension provides smaller vector space, thus the information it can accommodate is "shallower" than that of a higher dimension. This results in a decreased capability of accommodating the information required as well as ensuring the orthogonality of randomly generated HVs (which is critical for HDC). However, increasing dimensions will not always guarantee an accuracy increase. In most cases, detection accuracy saturates when dimensions reaches 15,000. Particularly, further increasing dimension to 30,000 will even cause slight accuracy drops for certain configurations due to dimensions that are excessively high for this application to effectively train.

As to the data types, we quantize the associative memory of **GuardHD** from 32-bit float to different integer types including binary, 8-bit and 16-bit signed integers. From Fig. 9, we can observe that quantizing the **GuardHD** from float to 32-bit integer does not arouse noticeable accuracy degradation. However, the 8-bit and binary integer **GuardHD** suffer more significant accuracy downgrade, particularly for binary where accuracy decreased to less than 70% which is barely acceptable for accurately detect adversarial samples.

To further explain the effect of the configurations, we present a deeper analysis on the HDC model performance by introducing an additional evaluation metric: the **detection confidence**. The detection confidence in **GuardHD** is defined as the normalized difference of output similarity between the correct class and the incorrect class given a set of samples. The detection confidence ranges from -1.0 to 1.0 where, 1). a positive sign means a correct prediction or vice versa, and 2) a higher absolute value means the HDC model is in greater confidence on making this prediction.

To not occupy excessive space. We randomly selected 50 samples from the test set of the MASK dataset and plot their detection confidences under float and binary data type in Fig. 10 as a case study. We can observe that when quantizing from floating point into binary, although some of the samples experience an increase in confidence, more samples, particularly those at the borderline (correct prediction with low confidence) are switching to incorrect predictions, which essentially causes the degradation of accuracy.

This is reasonable because when we quantize the model from data types with more bits such as floating point to binary which only occupies one bit, there is a significant loss of data. Specifically, for an HV with a dimension of 10,000 in 32-bit data, we are able to obtain a resolution of $2^{32^{10000}}$, while for a binary HV with the same dimension, the resolution is reduced to only $2^{1^{10000}}$, which is a $2^{32}\times$ smaller space. Such reduction on resolution can drastically degrade the volume of information that an HV can accommodate, impairing or even destroying the orthogonality required to present information from different modalities particularly when the initial item memory is generated as well as the sample HVs are encoded. In **GuardHD**, this is reflected as reduced confidence and even prediction errors during the detection on input samples as shown in Fig. 10.

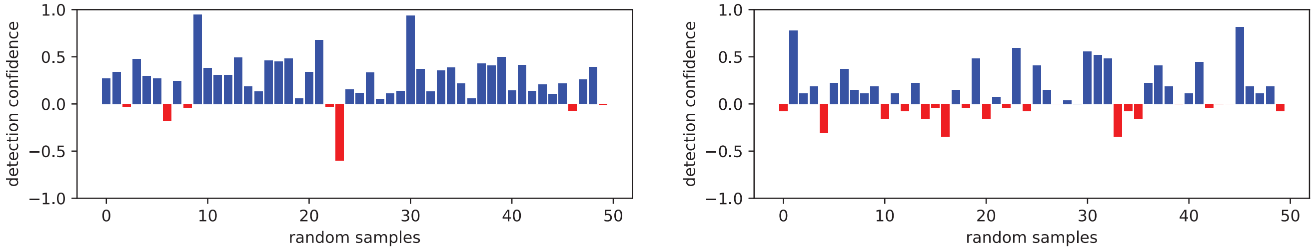
TABLE VI
THE DETECTION ACCURACY OF **GUARDHD** UNDER "CROSS-CHECKING" SCENARIOS.

train/test set	MNIST	Medical MNIST	MASK	FACE
black-box/gray-box	0.856	0.936	0.876	0.878
gray-box/black-box	0.9	0.945	0.906	0.881

C. Discussion on Attack Transferability Across Scenarios

In realistic implementations, the sources of the adversarial attack samples and how they are generated (black-box or gray-box) are usually unknown. The attacker may also face different attack settings and the available resources to initialize attacks can thus differ. For example, one attacker can possibly leverage side-channels to obtain information of associative memory to enable grey-box scenario of a specific architecture such as FPGA, yet such side-channel may not present in another architecture such as in-memory computing devices [31]. Another use-case is the emerging federated learning while an HDC model can be distributed to different devices with heterogeneous architectures [52], [53], therefore the scenario of initiating attacks may also vary. While it is impossible and also impertinent to exhaustively list the details of the scenarios such as how to enable vulnerabilities like side-channels, or how federated learning can be implemented in HDC, for a more realistic evaluation we still check the transferability of **GuardHD** by "cross-checking", i.e., training with samples from one scenario and evaluating with another.

In Table VI, we show the detection accuracy of **GuardHD** under two cross-checking scenarios with samples from different sources: training with gray-box samples and evaluating with black-box samples and, training with black-box samples and evaluating with gray-box samples. For some applications such as MNIST and Medical MNIST, the cross-checking detection accuracy of **GuardHD** is 2% – 10% less compared with scenarios with samples from the same source in Table V. However, even trained using samples from a single



(a) HVs in floating point

(b) HVs quantized into binary

Fig. 10. Detection confidence of **GuardHD** with 50 random samples under floating point and binary data type.

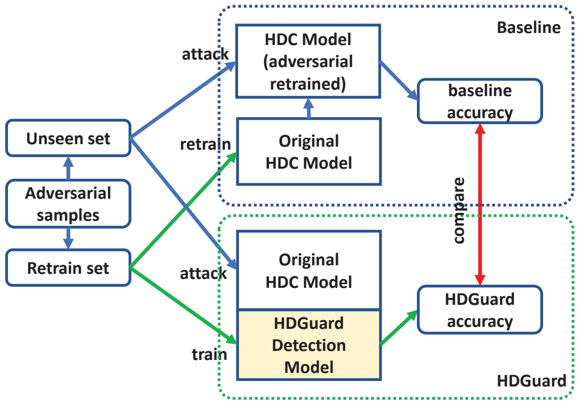


Fig. 11. Comparison between **GuardHD** and the baseline on detection and defense methodologies.

source of adversarial sample (either gray-box or black-box), **GuardHD** is still able to main higher than 85% accuracy on detecting adversarial from the other **unseen** sources. Interestingly, for FACE application, the detection accuracy under cross-checking scenarios becomes even higher to around 88%.

In addition, We notice that if we train **GuardHD** using the gray-box samples, **GuardHD** can perform better compared with training using the black-box samples by 2% – 7%. This indicate **GuardHD** is flexibly transferable in realistic implementations where adversarial samples can come from diverse types of unknown sources.

D. Overhead of *GuardHD*

We also analyze the overhead from the added **GuardHD** detection memory in two aspects: memory footprint (space complexity) and computation (time complexity). We assume that in the original HDC model, the item memory has N HVs using data width of W_{im}, associative memory has C class HVs using data width of W_{am} and all the HVs across the system are in the dimension of D.

The space complexity of the existing HDC model is $O(D(NW_{im} + CW_{am}))$ while the **GuardHD** space complexity is only $O(DW_{am})$. According to our analysis, the **GuardHD** overhead is less than 10% of the original HDC model and with quantization, the overhead can be further reduced to around 5%.

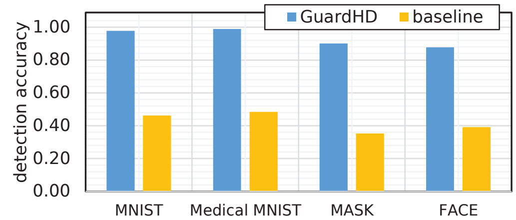


Fig. 12. **GuardHD** vs. adversarial training baseline in detection accuracy. **GuardHD** is up to 55% higher than the baseline.

As to time complexity, for HDC in general, encoding is the most computational intensive phase which accounts for 74% – 79% of the training and inference time [54]. In our applications, specifically, the time complexity of encoding is $O(DN^2)$ while for inference the **GuardHD** complexity is only $O(D)$. Across the four applications, time overhead on **GuardHD** is also less than 10% of the original HDC model. In short, **GuardHD** can enable over 80% accuracy for detecting adversarial attacks, at the expense of less than 10% overhead on both space and time.

E. *GuardHD* vs. Adversarial Training

The majority of the state-of-the-art adversarial detection and existing defense methods for HDC models are based on adversarial training, i.e., the generated adversarial samples are used for (re-)training the model [14], [15]. According to this concept, we implement a comparative study between **GuardHD** and such baseline, where the two detection and defense methods use the same set of adversarial samples generated, as illustrated in Fig. 11. **GuardHD** uses the adversarial samples for training the detection model while the baseline uses them for retraining the original HDC model. We then use another set of unseen adversarial samples to attack and observe the detection accuracy of **GuardHD** and the baseline. Note that existing works on adversarial training for HDC usually use attack success rate or accuracy as the metric for evaluation [40], [55]. For fair comparison with **GuardHD**, we also use detection accuracy in the place of attack success rate or attack accuracy as the metric for the adversarial training baseline.

In Fig. 12, we notice that the baseline detection and defense method that is based on retraining can only successfully identify 40% – 60% of all the attacks while **GuardHD** can consistently defend more than 80% of the attacks. Additionally, retraining based methods can potentially induce accuracy drops as adversarial samples incorporated in retraining are actually considered as noise or perturbation to the original dataset [14], which is also a common drawback of adversarial training on other machine learning algorithms such as deep neural networks [56]. A significant qualitative advantage of **GuardHD** is that **GuardHD** does not modify the existing HDC model, i.e., **GuardHD** brings less significant accuracy degradation compared with the adversarial training based methods.

VIII. Discussion and Future Works

While **TestHD** considers the use of L1 and L2 distance as the quality metrics of adversarial samples generated, it is possible to use other quality metrics. For example, generative models embrace more sophisticated metrics such as Inception Score [57] and Frechet Inception Distance (FID) [58] which align better with human judgement. However, such metrics rely on deep network models which can potentially occupy extremely high overhead and, how such metrics can transfer from content generation to adversarial generation is also outstanding.

Currently, **TestHD** provides two high-level budget metrics: the number of maximum allowed iterations of fuzzing and the generation time to indicate the cost of initiating the adversarial attacks. Depending on the specific generation strategies and computing resources, those metrics may not completely reflect the actual resources required particularly considering the budgets from the attacker can vary under different scenarios. Therefore, it is a potential future direction to explore if there are metrics at a finer granularity to more accurately assess the resources needed for adversarial attack under varying budgets.

While we compare **GuardHD** with adversarial training-based defense methods, there are more defense strategies that can be potentially applied to defend HDC models. For example, one of the successful methods for adversarial defense in DNN community is feature squeezing which aims to reduce the space available to an adversary by "squeezing out" unnecessary input features [59]. However, as those methods are originally designed and implemented for DNNs, how to transfer to the HDC domain will still need further study.

IX. Conclusion

As the emerging brain inspired hyperdimensional computing (HDC) is increasingly applied to security critical domains and applications, it is found vulnerable to attacks from adversarial input samples. In this paper we propose to test and enhance the robustness of HDC models against adversarial attacks by developing **TestHD** and **GuardHD**. **TestHD** is a highly-automated and scalable testing approach based on the differential fuzz testing principles. **TestHD** can iteratively mutates inputs to generate adversarial inputs to expose incorrect behaviors of HDC models, under either (unguided) black-box or (guided) gray-box scenarios. **GuardHD** is an approach for defending HDC against adversarial attacks. **GuardHD** performs binary classification within HDC model to detect adversarial data, which can be integrated to the existing HDC models with negligible overhead. We evaluate **TestHD** and **GuardHD** on 4 datasets and 5 adversarial attack scenarios with 6 adversarial generation strategies and 2 defense mechanisms. Experimental results also show that **GuardHD** is able to classify between benign and adversarial input samples with accuracy over 90% on average, surpassing adversarial-training baseline method by up to 55%.

Acknowledgments

This work was partially supported by NSF grant #2202310. 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] L. Ge *et al.*, "Classification using hyperdimensional computing: A review," *IEEE Circuits and Systems Magazine*, 2020.

[2] A. Rahimi *et al.*, "A robust and energy-efficient classifier using brain-inspired hyperdimensional computing," in *ISLPED*, 2016.

[3] M. Hersche, E. M. Rella, A. Di Mauro, L. Benini, and A. Rahimi, "Integrating event-based dynamic vision sensors with sparse hyperdimensional computing: a low-power accelerator with online learning capability," in *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*, pp. 169–174, 2020.

[4] A. Rahimi *et al.*, "Hyperdimensional computing for blind and one-shot classification of eeg error-related potentials," *Mobile Networks and Applications*, 2017.

[5] A. Moin *et al.*, "An emg gesture recognition system with flexible high-density sensors and brain-inspired high-dimensional classifier," in *ISCAS*, 2018.

[6] Y. Kim *et al.*, "Geniehd: efficient dna pattern matching accelerator using hyperdimensional computing," in *DATE*, 2020.

[7] C. Szegedy *et al.*, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[8] J. Yu, S. Duan, and X. Ye, "A white-box testing for deep neural networks based on neuron coverage," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[9] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[10] O. Russakovsky *et al.*, "Imagenet large scale visual recognition challenge," *IJCV*, 2015.

[11] S. Wang *et al.*, "Exploring causes and effects of automated vehicle disengagement using statistical modeling and classification tree based on field test data," *Accident Analysis & Prevention*, 2019.

[12] A. C. Madrigal, "Inside waymo's secret world for training self-driving cars," *The Atlantic*, 2017.

[13] D. Lowd and C. Meek, "Adversarial learning," in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pp. 641–647, 2005.

[14] F. Yang and S. Ren, "On the vulnerability of hyperdimensional computing-based classifiers to adversarial attacks," in *International Conference on Network and System Security*, pp. 371–387, Springer, 2020.

[15] D. Ma, J. Guo, Y. Jiang, and X. Jiao, "Hdtest: Differential fuzz testing of brain-inspired hyperdimensional computing," in *58th 2021 ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2021.

[16] J. Liang *et al.*, "Deepfuzzer: Accelerated deep greybox fuzzing," *IEEE TDSC*, 2019.

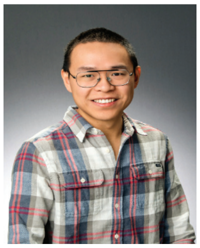
[17] W. M. McKeeman, "Differential testing for software," *Digital Technical Journal*, 1998.

[18] R. Thapa, D. Ma, and X. Jiao, "Hdxplore: Automated blackbox testing of brain-inspired hyperdimensional computing," *arXiv preprint arXiv:2105.12770*, 2021.

[19] P. Kanerva, "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors," *Cognitive computation*, vol. 1, no. 2, pp. 139–159, 2009.
[20] D. Kleyko, D. A. Rachkovskij, E. Osipov, and A. Rahim, "A survey on hyperdimensional computing aka vector symbolic architectures, part ii: Applications, cognitive models, and challenges," *arXiv preprint arXiv:2112.15424*, 2021.
[21] A. Burrello, K. Schindler, L. Benini, and A. Rahimi, "Hyperdimensional computing with local binary patterns: one-shot learning of seizure onset and identification of ictogenic brain regions using short-time ieeg recordings," *IEEE Transactions on Biomedical Engineering*, vol. 67, no. 2, pp. 601–613, 2019.
[22] A. Rahimi *et al.*, "Hyperdimensional biosignal processing: A case study for emg-based hand gesture recognition," in *ICRC*, 2016.
[23] M. Imani *et al.*, "Voicehd: Hyperdimensional computing for efficient speech recognition," in *2017 IEEE International Conference on Rebooting Computing (ICRC)*, IEEE, 2017.
[24] M. Imani, T. Nassar, A. Rahimi, and T. Rosing, "Hdna: Energy-efficient dna sequencing using hyperdimensional computing," in *2018 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*, pp. 271–274, IEEE, 2018.
[25] Y. Yao, W. Liu, G. Zhang, and W. Hu, "Radar-based human activity recognition using hyperdimensional computing," *IEEE Transactions on Microwave Theory and Techniques*, 2021.
[26] J. Liu *et al.*, "Hdc-im: Hyperdimensional computing in-memory architecture based on rram," in *ICECS*, 2019.
[27] M. Schmuck *et al.*, "Hardware optimizations of dense binary hyperdimensional computing: Rematerialization of hypervectors, binarized bundling, and combinational associative memory," *JETC*, 2019.
[28] T. F. Wu *et al.*, "Brain-inspired computing exploiting carbon nanotube fets and resistive ram: Hyperdimensional computing case study," in *ISSCC*, 2018.
[29] W. Chen and H. Li, "Adversarial attacks on voice recognition based on hyper dimensional computing," *Journal of Signal Processing Systems*, pp. 1–10, 2021.
[30] A. Hérnandez-Cano, R. Cammarota, and M. Imani, "Prid: Model inversion privacy attacks in hyperdimensional learning systems," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 553–558, IEEE, 2021.
[31] S. Duan, S. Ren, and X. Xu, "Hdlock: Exploiting privileged encoding to protect hyperdimensional computing models against ip stealing," *arXiv preprint arXiv:2203.09681*, 2022.
[32] A. Nguyen *et al.*, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *CVPR*, 2015.
[33] C. Xie, Y. Wu, L. v. d. Maaten, A. L. Yuille, and K. He, "Feature denoising for improving adversarial robustness," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 501–509, 2019.
[34] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu, "Defense against adversarial attacks using high-level representation guided denoiser," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1778–1787, 2018.
[35] Y. Ruan and J. Dai, "Twinnet: A double sub-network framework for detecting universal adversarial perturbations," *Future Internet*, vol. 10, no. 3, p. 26, 2018.
[36] J. Lu, T. Issaranon, and D. Forsyth, "Safetynet: Detecting and rejecting adversarial examples robustly," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 446–454, 2017.
[37] N. Chattopadhyay, S. Chatterjee, and A. Chattopadhyay, "Robustness against adversarial attacks using dimensionality," in *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pp. 226–241, Springer, 2021.
[38] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," *arXiv preprint arXiv:1705.07204*, 2017.
[39] A. Shafahi, M. Najibi, A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, "Adversarial training for free!," *arXiv preprint arXiv:1904.12843*, 2019.
[40] F. Yang and S. Ren, "Adversarial attacks on brain-inspired hyperdimensional computing-based classifiers," *arXiv preprint arXiv:2006.05594*, 2020.
[41] J. Guo *et al.*, "Dlfuzz: Differential fuzzing testing of deep learning systems," in *ESEC/FSE*, 2018.
[42] K. Pei, Y. Cao, J. Yang, and S. Jana, "Deepxplore: Automated whitebox testing of deep learning systems," in *proceedings of the 26th Symposium on Operating Systems Principles*, pp. 1–18, 2017.
[43] B. Khaleghi, J. Kang, H. Xu, J. Morris, and T. Rosing, "Generic: highly efficient learning engine on edge using hyperdimensional computing," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pp. 1117–1122, 2022.
[44] P. Poduval, Z. Zou, H. Najafi, H. Homayoun, and M. Imani, "Stochd: Stochastic hyperdimensional system for efficient and robust learning from raw data," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 1195–1200, IEEE, 2021.
[45] M. Imani, S. Bosch, S. Datta, S. Ramakrishna, S. Salamat, J. M. Rabaey, and T. Rosing, "Quanthd: A quantization framework for hyperdimensional computing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 10, pp. 2268–2278, 2019.
[46] M. Imani, X. Yin, J. Messerly, S. Gupta, M. Niemier, X. S. Hu, and T. Rosing, "Searchd: A memory-centric hyperdimensional computing with stochastic training," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 10, pp. 2422–2433, 2019.
[47] Y. LeCun *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, 1998.
[48] apolanco3225, "Medical mnist classification." https://github.com/apolanco3225/Medical-MNIST-Classification, 2017.
[49] S. Asif, Y. Wenhui, Y. Tao, S. Jinhai, and K. Amjad, "Real time face mask detection system using transfer learning with machine learning method in the era of covid-19 pandemic," in *2021 4th International Conference on Artificial Intelligence and Big Data (ICAIBD)*, pp. 70–75, IEEE, 2021.
[50] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.
[51] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.
[52] C.-Y. Hsieh, Y.-C. Chuang, and A.-Y. A. Wu, "Fl-hdc: Hyperdimensional computing design for the application of federated learning," in *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pp. 1–5, IEEE, 2021.
[53] Q. Zhao, K. Lee, J. Liu, M. Huzaifa, X. Yu, and T. Rosing, "Fedhd: federated learning with hyperdimensional computing," in *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, pp. 791–793, 2022.
[54] M. Imani, J. Morris, J. Messerly, H. Shu, Y. Deng, and T. Rosing, "Bric: Locality-based encoding for energy-efficient brain-inspired hyperdimensional computing," in *Proceedings of the 56th Annual Design Automation Conference 2019*, pp. 1–6, 2019.
[55] H. Moraliyage, S. Kahawala, D. De Silva, and D. Alahakoon, "Evaluating the adversarial robustness of text classifiers in hyperdimensional computing," in *2022 15th International Conference on Human System Interaction (HSI)*, pp. 1–8, IEEE, 2022.
[56] M. Alzantot, Y. Sharma, S. Chakraborty, H. Zhang, C.-J. Hsieh, and M. B. Srivastava, "Genattack: Practical black-box attacks with gradient-free optimization," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1111–1119, 2019.
[57] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *Advances in neural information processing systems*, vol. 29, 2016.
[58] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *Advances in neural information processing systems*, vol. 30, 2017.
[59] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," *Network and Distributed Systems Security Symposium (NDSS)*, 2018.

Dongning Ma (S'19) is currently a Ph.D. candidate in the Department of Electrical and Computer Engineering at Villanova University, USA. He obtained his Bachelor of Engineering degree in automation from School of Advanced Engineering in University of Science and Technology Beijing (USTB), China. His research interests include approximate computing, bio-inspired computing, computational intelligence and bio-informatics.

Tajana Šimunić Rosing (F'05) received the M.S. degree in engineering management concurrently with the Ph.D. degree from Stanford University, Stanford, CA, USA, in 2001 with the Ph.D. topic "Dynamic Management of Power Consumption." She is a Professor, a Holder of the Fratamico Endowed Chair, and the Director of System Energy Efficiency Laboratory, University of California at San Diego, La Jolla, CA, USA. From 1998 to 2005, she was a full-time Research Scientist with HP Labs, Palo Alto, CA, USA, while also leading research efforts with Stanford University, Stanford, CA, USA. She was a Senior Design Engineer with Altera Corporation, San Jose, CA, USA. She is leading a number of projects, including efforts funded by DARPA/SRC JUMP CRISP program with focus on design of accelerators for analysis of big data, DARPA and NSF funded projects on hyperdimensional computing and SRC funded project on IoT system reliability and maintainability. Her current research interests include energy efficient computing, cyber-physical, and distributed systems.



Xun Jiao is Assistant Professor in the Department of Electrical and Computer Engineering at Villanova University, USA. He received the dual bachelor's degree from the joint program of Beijing University of Posts and Telecommunications, China and Queen Mary University of London, UK in 2013. He received the Ph.D. degree from the Department of Computer Science and Engineering from University of California, San Diego in 2018. His research interests include embedded systems, design automation, and brain-inspired computing. He is an associate editor of IEEE TCAD, and a TPC member of DAC, GLSVLSI, and LCTES.