Invited Paper: Hyperdimensional Computing for Resilient Edge Learning

Hamza Errahmouni Barkam[1], SungHeon Eavn Jeon[1], Sanggeon Yun[1], Calvin Yeung[1]
Zhuowen Zou[1], Xun Jiao[2], Narayan Srinivasa[3], and Mohsen Imani[1,†]
[1]University of California Irvine, [2]Vilanova University, [3]Intel Labs
[†]Correspondance: m.imani@uci.edu

2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD) | 979-8-3503-2225-5/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICCAD57390.2023.10323671

***Abstract*—Recent strides in deep learning have yielded impressive practical applications such as autonomous driving, natural language processing, and graph reasoning. However, the susceptibility of deep learning models to subtle input variations, which stems from device imperfections and non-idealities, or adversarial attacks on edge devices, presents a critical challenge. These vulnerabilities hold dual significance—security concerns in critical applications and insights into human-machine sensory alignment. Efforts to enhance model robustness encounter resource constraints in the edge and the black box nature of neural networks, hindering their deployment on edge devices. This paper focuses on algorithmic adaptations inspired by the human brain to address these challenges. Hyper Dimensional Computing (HDC), rooted in neural principles, replicates brain functions while enabling efficient, noise-tolerant computation. HDC leverages high-dimensional vectors to encode information, seamlessly blending learning and memory functions. Its transparency empowers practitioners, enhancing both robustness and understanding of deployed models. In this paper, we introduce the first comprehensive study that compares the robustness of HDC to white-box malicious attacks to that of deep neural network (DNN) models and the first HDC gradient-based attack in the literature. We develop a framework that enables HDC models to generate gradient-based adversarial examples using state-of-the-art techniques applied to DNNs. Our evaluation shows that our HDC model provides, on average, 19.9% higher robustness than DNNs to adversarial samples and up to 90% robustness improvement against random noise on the weights of the model compared to the DNN.**

I. Introduction

Modern deep learning has achieved impressive breakthroughs in practical applications fueled by large datasets. For instance, existing deep learning models have demonstrated remarkable accuracy in tasks like extensive image categorization [1]. However, the susceptibility of deep learning algorithms to subtle input alterations poses a significant challenge, and these perturbations can arise from random noise inherent to device imperfections or from intentional attacks exploiting the vulnerabilities of edge devices. Such devices, often constrained by their limited computational capacities, are particularly susceptible to such vulnerabilities, which can further limit the kinds of deployable models.

These vulnerabilities that deep neural networks exhibit against perturbations are well-documented as "adversarial attacks" [2]. Comprehending these adversarial perturbations holds a two-fold significance [3]: firstly, it pertains to the security of deployed machine learning algorithms, particularly in safety-critical applications like autonomous vehicles; secondly, it serves to bridge the gap between human and machine sensory information processing, which can guide the development of robust, brain-inspired learning paradigms.

However, ongoing efforts to enhance the accuracy and resilience of deep learning models against adversarial attacks are not without limitations. On the one hand, executing deep learning algorithms requires considerable computational resources and storage capacities that often exceed what current edge devices can provide [4]. Consequently, many devices lack the capability for on-device security monitoring, necessitating the transmission of data to remote cloud servers for analysis. This practice, though convenient, introduces concerns related to scalability, security, and data privacy [5]. Furthermore, deep learning models, unlike the human brain, inherently lack the robustness needed to handle adversarial perturbations.

Acknowledging a significant roadblock in deploying neural networks, particularly in edge device scenarios, pertains to their often-opaque nature. Neural networks, often called "black boxes," introduce challenges in comprehending, interpreting, and subsequently enhancing their performance. This limitation curtails meaningful enhancements and thorough investigations into their behavior, particularly when embedded in resource-constrained environments like edge devices.

To address these challenges and achieve real-time performance combined with robustness in AI models, we propose algorithmic adjustments inspired by the human brain. In this pursuit, we advocate for adopting Hyper Dimensional Computing (HDC), a groundbreaking paradigm firmly rooted in neural principles [6]–[9]. HDC has garnered noteworthy acclaim for its adept replication of essential brain functions, all while providing a platform for efficient and noise-tolerant computation. That is the main reason why HDC emerges as a promising solution, not only offering resilience against adversarial perturbations but also delivering heightened transparency and interpretability [10]–[13]. This inherent transparency empowers practitioners to glean insights into the inner workings of deployed models, thereby paving the path for more informed and effective improvements.

Central to the HDC framework lies a profound insight derived from the human brain's information-processing mechanisms, which hinge on intricate high-dimensional representations. Within the HDC framework, entities are encoded using elaborate high-dimensional vectors referred to as "hypervectors," each painstakingly constructed from an array of thousands of elements [14]. Notably, HDC seamlessly amalgamates learning capabilities with memory functions that closely mirror the operations of human memory systems. This innovative approach effectively mimics core memory functions through the execution of vector-based operations, thereby conferring both practical feasibility and mathematical rigor to the paradigm.

Recent strides have showcased HDC's advantages over alternative learning methods: (1) its exceptional parallelism and suitability for real-time on-device learning scenarios [15]–[20]; (2) its capability to achieve effective learning with minimal sample usage [21]; and (3) its robustness against noise and data corruption (OnlineHD). HDC

has demonstrated superior accuracy with fewer training instances in comparison to support vector machines (SVMs), gradient boosting, and deep learning models [21], [22]. By bridging the gap between neural principles and real-world applications, HDC opens avenues for achieving both human-like robustness and a deeper grasp of the model's behavior. This momentum propels the field toward more secure, knowledgeable deployments in edge computing environments.

On the backbone of these promising advancements, an important part of the research must focus on HDC's vulnerability to perturbations in several scenarios. HDC algorithms have recently found utility in security-critical industrial applications underscoring the importance of thoroughly exploring the susceptibility of HDC algorithms to adversarial samples. The main contributions of the paper are listed as follows:

- Comprehensive comparative study on the robustness of HDC and DNN models versus perturbations on the weights of the model given two of the most comment non-idealities on the edge.
- We develop a novel framework that enables HDC models to generate gradient-based adversarial examples. We define a loss function and back-propagation on the HDC model, which enables us to generate adversarial samples using state-of-the-art attack methods: Fast Gradient Sign Method [23], Jacobian-based Saliency Map Attack [24], and DeepFool [25]. Finally, we measure their effectiveness against both HDC and DNN.
- We study the effect of HDC hyperparameters such as encoding and dimensionality on the robustness of the model. Finally, we design defense mechanisms to protect HDC models against adversarial samples, such as data pre-processing and adversarial training on gradient-based samples. Our study also indicates that the defense mechanisms can have significant effectiveness against some specific attacks (for example, 24% enhancement on DeepFool), but further work is still needed in this area.

We evaluate our solution on several scenarios, such as three popular image classification datasets, MNIST, E-MNIST, and FMNIST. Our results indicate that the HDC model provides, on average, 19.9% higher robustness than DNNs to adversarial samples. Understanding that adversarial samples are often considered universal, the adversarial samples generated by DNN are not successful in attacking the HDC model (attack success rate under 5%), which proves HDC provides inherent robustness against noise.

II. Background

A. Hyperdimensional Computing

Hyper-dimensional Computing (HDC) emerged from theoretical neuroscience as a short-term human memory model [14], [26]. The cerebellum is an area in the brain that plays a significant role in cognitive functions. Each dimension of the hypervector models a neuron's function at an abstract level [27]. When we generate hypervectors on the high-dimensional space, our encoding works in a way where there is a huge number of nearly orthogonal hypervectors. This enables us to combine such hypervectors using well-defined operations while keeping the information of the two with high probability. There are three main operators on HDC:

- **(1) Bundling:** Describes the memorization of a set of hypervectors, which results in a single hypervector that shares similarities with all its elements. For two hypervectors V_1 and V_2, bundling corresponds to the element-wise sum: $R = V_1 + V_2$.
- **(2) Binding:** Represents the association of multiple dissimilar hypervectors (e.g., V_1, V_2), which generates a single hypervector ($R = V_1 * V_2$). The bound hypervector is a new object in HDC space which is orthogonal to each one of its components ($\delta(R, V_1) \simeq 0$ and $\delta(R, V_2) \simeq 0$).
- **(3) Permutation:** Is an operator that encodes order. It consists of applying a single rotational shift. The permuted hypervector will be dissimilar to its original hypervector ($\delta(V_1, \rho V_1) \simeq 0$). We apply permutation as many times as the position of the element in the sequence, i.e., BCDA can be encoded as $V_B * \rho V_C * \rho^2 V_D * \rho^3 V_A$.

There is not a lot of literature on HDC against adversarial attacks. The first publication [28] demonstrates that HDC is vulnerable to black-box attacks, specifically genetic algorithms, and proposes negative training as a defense technique. However, it has the following limitations: (1) the study is limited only to binary hypervectors, (2) It does not cover effective and popular white-box attacks such as gradient-based attacks, and (3) it does not compare its results with traditional neural networks. Subsequent publications, such as [22], [29], [30], study the robustness of HDC against black-box attacks in different domains, such as voice recognition. However, they fail to show gradient-based methods to generate the samples (white-box) or compare them to DNNs.

B. Adversarial Attacks

An intriguing discovery in 2014 [31] showed the weakness of DNN models to adversarial attacks. Adversarial samples are any perturbation to the original input that can change the predicted label and, more often, cause the model to have high confidence in the wrong class. DNN adversarial samples are often considered universal since the same instance can trick multiple classifiers. These results prompted many researchers to gain interest in this topic and investigate how to defend against adversarial attacks [32]–[34].

Adversarial research begins by defining the threat model, which consists of describing the goal of the attacker and the amount of knowledge he has of the model. Among all types of threat models, white-box attacks are those where the adversary has full knowledge of the model and all its parameters. White-box attacks such as FGSM [23], JSMA [24] and Deep Fool [32] have gained traction. They have become widely used to test the robustness of models due to their ability to generate highly successful adversarial inputs in a very efficient manner using the gradient of the model's loss function.

Such research advancements and discoveries of the vulnerabilities of learning models have also caused the appearance of defense strategies that increase the robustness of models against adversarial attacks. The most intuitive technique against attacks proposed is adversarial training [23], which consists of retraining the model on generated negative samples to improve robustness without losing much accuracy. Many new defense strategies have appeared, and one showing tremendous potential is pre-processing images to eliminate noise, which many adversarial algorithms do not account for [35].

Unlike existing DNN models, which are gradient-based, HDC has a different learning methodology as it is a pattern-based computational model. Therefore, adversarial sample generators and defense mechanisms developed for DNN may not be effective against HDC. In this paper, we explore the capability of the HDC model to generate adversarial samples. Accordingly, we define new defense mechanisms that are well-developed for it.

III. Hyperdimensional Classification

At training time, the first step is to encode data points into high-dimensional space. The encoded hypervectors are combined via brain-like operations to train a suitable learning model, resulting in a

hypervector representation for each class. The similarity between an encoded query hypervector and the various class hypervectors is computed at inference time. Then, the class with the highest similarity is selected as the predicted class.

A. Hyperdimensional Encoding

The first step of HDC is an encoding module that maps each data point to high-dimensional space. HDC leverages different encoding methodologies according to the data types [15], [36]. The encoding process of our framework is characterized as Locality Preserving Encoding (LPE), which produces vector representations of points so that the inner product of the vectors reflects the relationship between the points in the original space. The encoded data should satisfy the common-sense principle: data points separate from each other in the original space should also be different in the hyperspace. All encoding methods are based on the fact that data points in the original space should also be different in high dimensions.

The encoding method we use exploits the kernel trick [37], [38]. The underlying idea of the kernel trick is that data, which may not be linearly separable in original dimensions, might be linearly separable if projected to higher dimensions. The Radial Basis Function or Gaussian Kernel $K(x,y) = e^{\frac{-||x-y||^2}{2\sigma^2}}$ is the most popular kernel.

The study in [37] showed that the dot product of two encoded inputs can efficiently approximate the Radial Basis Function (RBF) kernel, such that: $K(x,y) \approx z(x) \cdot z(y)$, where the encoding $z(\cdot)$ generally maps to high dimensional space. Figure 1 ⓐ shows the functionality of our encoding method. The proposed encoding method is inspired by the RBF kernel.

Let us consider an encoding that maps a feature vector $\mathbf{F} = (f_1, \ldots, f_n) \in \mathbb{R}^n$ to a hypervector at inference time $\mathbf{H} = (h_1, h_2, \ldots, h_D) \in \mathbb{R}^D$. We generate each dimension of the hypervector as follows:

$$h_i = \cos(\mathbf{B}_i \cdot \mathbf{F} + b) \times \sin(\mathbf{B}_i \cdot \mathbf{F}), \tag{1}$$

where $\mathbf{B}_i \sim \mathcal{N}(\vec{0}_n, I_n)$ and $b \sim \text{Unif}(0, 2\pi)$ (Figure 1 ⓑ). The random vectors $\{\mathbf{B}_1, \mathbf{B}_2, \cdots, \mathbf{B}_D\}$ can be generated once offline and can then be used for the rest of the classification task. Hypervectors generated as above precisely approximate the Gaussian kernel similarity via the dot product operation [37].

B. Training

The objective of the HDC training is to find the universal patterns from the training dataset that describe each one of the labels. During training, the encoded hypervectors are linearly combined to create a hypervector representation for each class (Figure 1 ⓒ). For each data point, we compute the similarity between it and all class hypervectors $\{\mathbf{C}_l\}$, $\delta(\mathbf{H}, \mathbf{C}_l)$. We then update the model based on the δ similarity and the model's correctness. For example, if the data point has a true label of l but the more similar to class l', we update the model as follows:

$$\mathbf{C}_l \leftarrow \mathbf{C}_l + \eta(1 - \delta_l) \times \mathbf{H} \qquad \mathbf{C}_{l'} \leftarrow \mathbf{C}_{l'} - \eta\delta_{l'} \times \mathbf{H}$$

where η represent the learning rate, and $\delta_l = \delta(\mathbf{H}, \mathbf{C}_l)$ and $\delta_{l'} = \delta(\mathbf{H}, \mathbf{C}_{l'})$ are the similarity of an encoded train data to true and incorrectly predicted classes. As such, we ensure that the model is updated based on how far a training data point is misclassified with the current model. Also, we provide separate coefficients for the true and miss-predicted labels, allowing us to update each class hypervector independently.

C. Inference

When deducing an inference, we encode and generate a query hypervector $\mathbf{H}$. We compute the similarity between the query hypervector and all class hypervectors. The model takes the class with maximum similarity as the predicted class for the query.

IV. HDC Adversarial Attack

In this section, we explore the robustness of HDC models to state-of-the-art adversarial attacks. Figure 1 shows an overview of our framework. All the existing attacks that we use in this paper require us to define a gradient over the HDC model with respect to some loss function. However, the existing HDC models are trained using simple operations (bundling, binding, and permutation) and do not rely on the optimization of a loss function.

Figure 1 shows the overview of our HDC model with holographic gradient-based computation. During inference, the model predicts the class based on the similarity of a query with all class hypervectors. We pass the similarities through an additional softmax layer. We define a loss function with the goal of changing the class similarity values in the desired direction. We retrieve an adversarial hypervector from the loss function (ⓓ), and then we go back to the original space through the activation function (ⓔ) and the encoding matrix (ⓕ), giving us the desired adversarial noise. Although backpropagation through the HDC model can be accurate since our attacks were successful using this framework, as explained in Section III-A, our encoding method exploits a periodic activation function and high-dimensional encoding matrix that generates quasi-orthogonal hypervectors and introduces non-linearity. Unlike neural networks, where backpropagation is necessary to the DNN framework, HDC does not require it. We generate adversarial samples to adapt to a white-box scenario. However, the perturbations are more perceptible, which shows the need to design a white-box attack catered to HDC mathematics, something we are planning to work on for future work.

A. Threat Model

Threat Modeling consists of identifying the security threats of the experiment, often characterized by describing the profile of the attacker and their objectives, the system vulnerable to the attack, and the access the attacker has to the system. Threat models are often found in two categories, white-box or black-box attacks. In black-box scenarios, the attackers have no knowledge of the architecture of the target model, so they generate adversarial samples by getting feedback from the predictions of the model by given input (queries). On the other hand, white-box models assume that the attacker has full access to the target model, often using the gradient of the input to generate adversarial samples. In our threat model, we consider an adversary/attacker as any subject that generates samples that can fool the classifier by causing erroneous predictions. We compare the robustness of the HDC and DNN models against three popular white-box attacks. Our goal is to minimize the accuracy of the model while generating subtle perturbations.

B. Fast Gradient Sign Method

The first white-box attack we use is the Fast Gradient Sign Method (FGSM), which consists of an algorithm that produces malicious samples from the gradient of the cost function relative to the inputs. Work in [23] is one of the most effective FGSM attacks on DNN models. We adapted FGSM by adding a loss function suitable to our framework in order to generate adversarial samples with HDC.

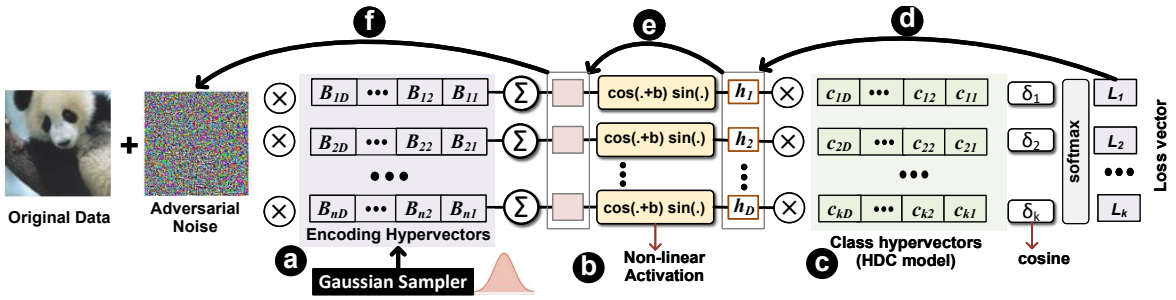

Fig. 1. Hyperdimensional computing with non-linear encoding. The backpropagation of the model could formal loss function.

In FGSM, the perturbations are calculated as

$$\eta = \epsilon \operatorname{sign}(\nabla_x J(\theta, x, y)), \tag{2}$$

where ϵ is the perturbation magnitude, θ are the model parameters, x is the input of the model, y is the target label, and J is the loss function. In the case of DNN, we take J to be the MSE loss function.

For HDC, we defined the encoding methodology on III-A. Let us define an encoded query $\mathcal{H}_x$. Since classification is defined as the maximum cosine similarity between class hypervectors $\mathcal{C}_\rangle$ and the query, our loss function is the cosine loss function:

$$J(\theta = \forall \mathcal{C}_i, x, y) = \frac{\mathcal{C}_i \cdot \mathcal{H}_x}{||\mathcal{C}_i||\ ||\mathcal{H}_x||} - 1 \tag{3}$$

Since we are applying FGSM, we get the gradient as $\nabla_x J(\theta = \forall \mathcal{C}_i, x, y)$, and then compute the perturbation by multiplying by epsilon.

C. Jacobian Based Saliency Map Attack

Jacobian-Based Saliency Map Attack (JSMA) is another gradient-based white-box method. Work in [24] proposed to use the gradient of the loss with each class label concerning every component of the input, i.e., Jacobian matrix, to extract the sensitivity direction. Then a saliency map is used to select the dimension which produces the maximum error using the following equation (1):

$$S^+(x_{(i)}, C_i) = \begin{cases} 0 \text{ if } \frac{\partial f_{C_i}(x)}{\partial x_{(i)}} < 0 \text{ or } \sum_{C' \neq C_i} \frac{\partial f_{C_i}(x)}{\partial x_{(i)}} > 0 \\ -\frac{\partial f_{C_i}(x)}{\partial x_{(i)}} \cdot \sum_{C' \neq C_i} \frac{\partial f_{C_i}(x)}{\partial x_{(i)}} \text{ otherwise} \end{cases} \tag{4}$$

Where $f_{C_i}(x)$ corresponds to the softmax probability for class C_i predicted by the victim model, i.e. $f_{C_i}(x) = \text{softmax}(\hat{y}(x))_i$, where $\hat{y}(x)$ is the output of the DNN (with no softmax layer). To adapt JMSA to the HDC framework, we have $f_{C_i}(x) = \text{softmax}\left(\frac{\mathcal{C}_i \cdot \mathcal{H}_x}{||\mathcal{C}_i||\ ||\mathcal{H}_x||}\right)$. Once we have the saliency map, we generate the noise samples by selecting the pixel positions that produce the maximum error. This consists of penalizing gradients associated with small probabilities to mitigate their influence (sensitivity).

The significant distinction between JSMA with FGSM is that it reduces the number of perturbations, making the adversarial examples far less detectable. But this comes at the expense of a higher computation cost. Also, JSMA is helpful for targeted misclassification attacks [39].

D. DeepFool Attack

DeepFool [25] is a recent white-box attack that is a simple and accurate method for computing the robustness of different classifiers to adversarial perturbations. DeepFool uses a model and an input image and outputs the minimal perturbation required to misclassify an image. We are considering the algorithm for multi-classification perturbation. We start from an input x_0 and the classifier f. Next, we define the perturbed image as the original image $x_i = x_0$ and iteratively perturb the data until the original label y_0 and the predicted label y_i are not matching.

Each iteration t begins by going over all the classes (C_i) and storing the minimum difference between the gradient of the original image and that of each one of the classes, and also the difference in outputs:

$$w'_{C_i} = \nabla f_{C_i}(x_i) - \nabla f_{y_0}(x_0) \tag{5}$$

$$f'_{C_i} = f_{C_i}(x_i) - f_{y_0}(x_0) \tag{6}$$

Given these values for every class, we compute the closest hyperplane for the input x_0 as:

$$\hat{l}(x_0) = \underset{C_i \neq y_0}{\operatorname{argmin}} \frac{|f_{C_i}(x_0) - f_{y_0}(x_0)|}{||w_{C_i} - w_{y_0}||_2} \tag{7}$$

Then, we derive the minimal vector that projects x onto the closest hyperplane from the previous step:

$$r_*(x_0) = \frac{|f_{\hat{l}(x_0)}(x_0) - f_{y_0}(x_0)|}{||w_{\hat{l}(x_0)} - w_{y_0}||_2^2}(w_{\hat{l}(x_0)} - w_{y_0}) \tag{8}$$

The last step of the iteration is adding the minimal perturbation to the image and checking if it is misclassified. If that is the case, the output will be the total perturbation, which is the sum of all perturbations. For HDC and DNN, the gradients are computed similarly to FGSM.

Figure 2 represents the effect of the attacks on FMNIST samples. The samples are generated using HDC and DNN models (epsilon of 0.01). We can observe how DeepFool is the best attack to hide the perturbation to the human eye, but we also notice how HDC yields more perturbation than DNN on FGSM using the same epsilon values.

V. Adversarial Defense

Many studies have been conducted on defense techniques for adversarial attacks against DNNs [40]. We have yet to see any effort toward making HDC learning models robust using novel techniques such as data filtering or adversarial training using white-box samples. In this section, we present defense techniques that focus on two mechanisms: (1) retraining the model on gradient-based generated adversarial samples; and (2) pre-processing input data by applying a noise reduction filter, a novel technique that, to our knowledge, has

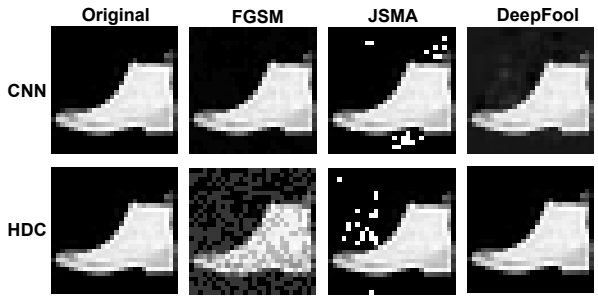

Fig. 2. Adversarial attacks using a DNN and HDC model on FMNIST.

never been applied to HDC. We then analyze how the accuracy varies for each defense technique and compare them to the original HDC classifier previously described.

A. Retraining on Adversarial Samples

We use Projected Gradient Descent (PGD) [41], another white-box attack method, to generate adversarial samples for retraining purposes. Compared to other attacks that aim to maximize the model's loss function, PGD imposes almost no constraints on resources in time and computation to find the perturbation with minimal magnitude; these samples thus provide more comprehensive coverage for defense.

B. Image Filtering

Image filtering tries to keep the classifier intact by pre-processing the input to reduce noise, which we consider to be any change in pixel values in an image. In image processing, additive noise is a problem that has been well studied, mostly done to recover the intensity and value of pixels. Consequently, several simple and effective techniques have been developed to solve this problem [42]. We implement average local pooling as our filtering method that computes the average for each patch of the pixels. To avoid detection, adversarial noise is typical of small magnitude. This technique effectively erases added noise.

VI. Evaluations

A. Experimental Setup

We developed a PyTorch-based code to implement Hyperdimensional classification along with adversarial sample generation, attack, and defense. Both DNN and HDC implementations were optimized to maximize performance by utilizing GPU resources.

Our model is extensively evaluated on three popular datasets: MNIST [43], an extended MNIST (EMNIST) [44] and Fashion-MNIST [45]. MNIST and EMNIST consist of handwritten digits and Fashion-MNIST is a dataset of clothing images. Each example is a 28×28 gray-scale image and both of them have a training set of 60,000 examples and a test set of 10,000 examples. We generated adversarial samples for both HDC (described in section IV) and DNN models, which specifically consist of a 3-layer Convolutional Neural Network.

B. Robustness of model against perturbations on the weights

Within this section, we focus on the intricate interplay of perturbations on model weights, a pivotal focus in the realm of AI. This approach encompasses the construction of weight histograms for classifiers, where significance is attributed to dominant values. Our investigation encompasses two prevalent perturbation scenarios encountered in edge and computing-in-memory environments. The first scenario, named "state shift probability," introduces symbol transitions similar to scaled Computing-In-Memory FeFET cells. The second

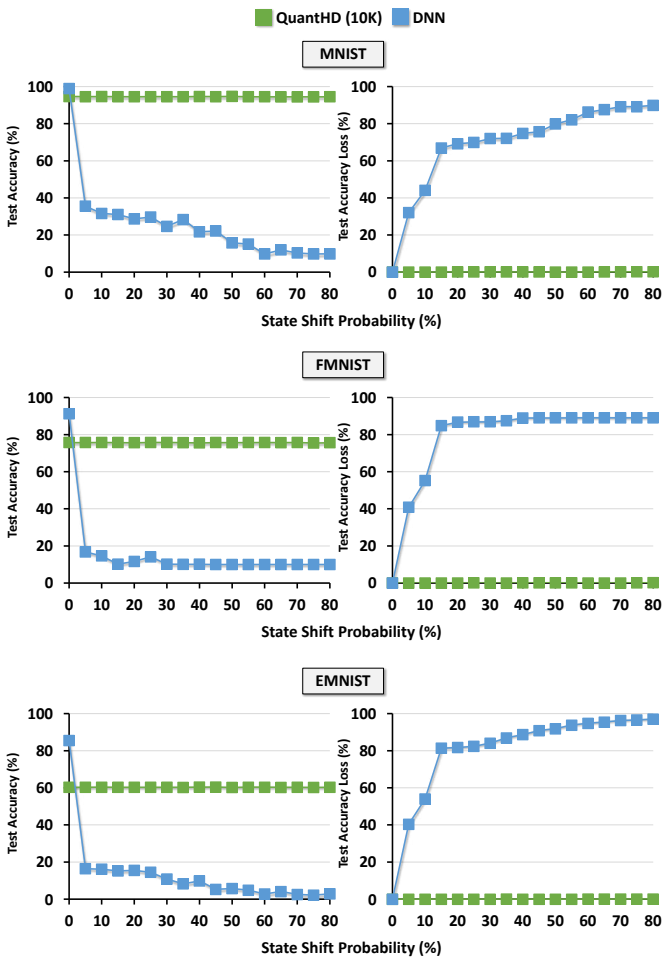

Fig. 3. Accuracy of Quantized version of HDC and DNNs when given different probabilities for the model to shift model weight values, often found in Computing in Memory based noise.

scenario involves random bit inversion. These scenarios mirror real-world challenges arising from emerging technologies. Our findings notably underscore HDC's exceptional superiority over DNN counterparts. Impressively, HDC's performance demonstrates only minimal accuracy degradation, as shown in Figures 3 and 4, emphasizing its robustness in the face of induced perturbations. Furthermore, HDC's training efficiency is still orders of magnitude higher compared to DNN, an achievement given HDC's intrinsic efficiency compared to DNN backpropagation.

We expanded our exploration by testing an HDC clustering model's resilience against noise contextualized on the MNIST dataset. The model being used is described in [46]. This evaluation fortifies the idea of HDC's natural robustness. Through this analysis, we examine the impact of perturbations on the model's performance in Figure 5. Intriguingly, the results support the idea of HDC's resilience, as minor variations are observed in the generated clusters, yet the underlying classification remains consistently accurate until 50% noise is considered. This reinforces HDC's adaptability and robustness across various scenarios, underlining its potential for deployment in a wide array of applications.

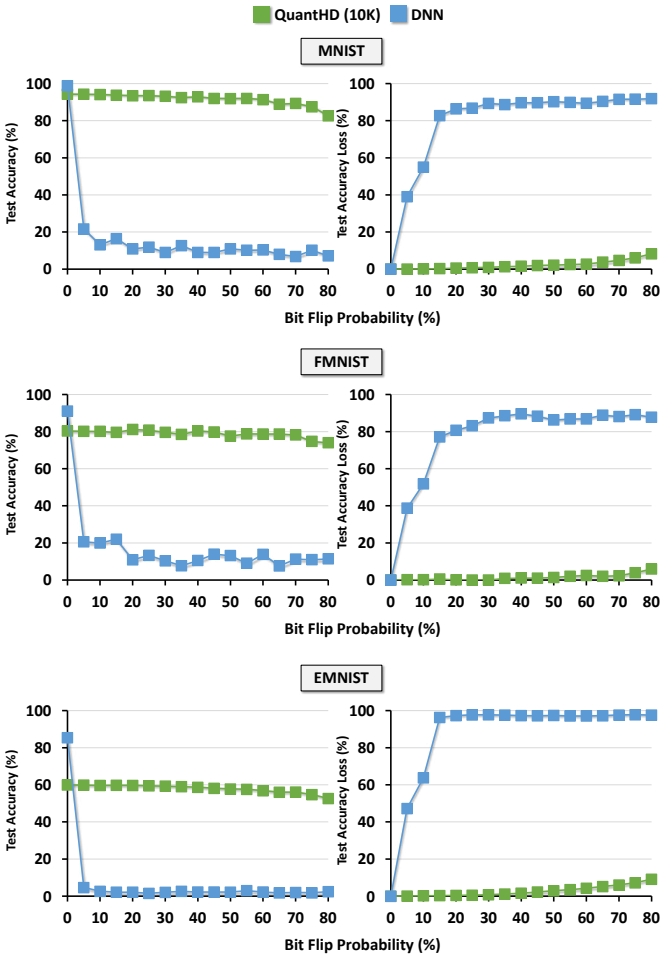

Fig. 4. Accuracy of Quantized version of HDC and DNNs when given different probabilities for the model to flip any bit of its model weight values, often considered for basic noisy scenarios.

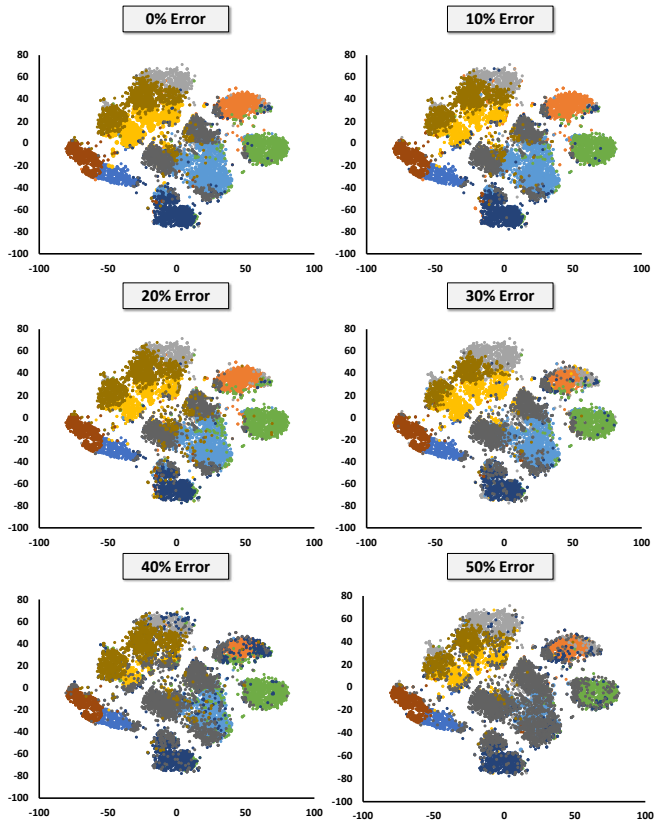

Fig. 5. Applying bit flip noise error on the HDC clustering model and showing the different classifications for several digits plotted using TSNE.

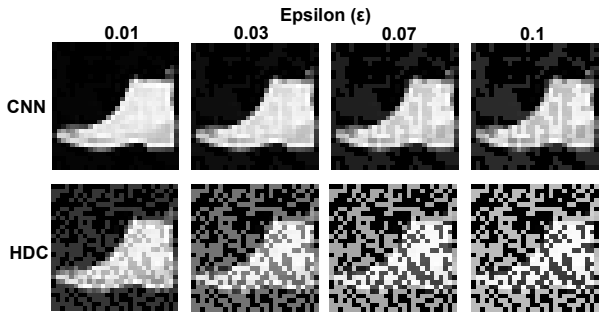

Fig. 6. DNN and HDC adversarial samples with FGSM varying epsilon.

C. HDC & DNN Attacks by FGSM

The goal of this experiment is to compare each model's malicious samples in order to have fairness. Figure 6 visually compares the adversarial samples generated by FGSM using different epsilon (ϵ) values. The figure shows that HDC backpropagation of the noise has more noticeable perturbations than DNN using the same epsilon values. This comes from our periodic activation, high dimensionality, and holographicness of our encoding process. In order to compare the robustness of DNN with HDC, we quantitatively compare perturbations in terms of noise for the adversarial images. Using the FGSM method, our results indicate that HDC adversarial samples with $\epsilon = 0.01$ have the equivalent perturbation to DNN using $\epsilon = 0.03$, and these will be the epsilon values used for fairness.

D. HD & DNN Robustness to Adversarial Attack

Figure 7a compares the robustness of HDC and DNN models to adversarial samples generated by different attack mechanisms. We report adversarial classification accuracy for all three datasets. Higher accuracy indicates higher robustness to adversarial perturbation. Our evaluation shows that DNN is highly vulnerable to adversarial samples generated by a DNN model. In contrast, HDC using our non-linear encoding provides natural robustness to adversarial attacks. For the example with the MNIST dataset and the same perturbation magnitude, our HDC model achieves 11.15%, 57.16% and 20.19% higher accuracy than DNN models using FGSM, Deep Fool, and JSMA attacks, respectively. Comparing different attack mechanisms, we observe that HDC has the highest sensitivity to DeepFool attacks. As explained in Section IV, HDC exploits non-linear and non-convex encoding methods, thus making gradient-based attacks relatively unsuccessful. In contrast, DeepFool exploits non-gradient-based sample generation which provides more success in attacking the HDC model.

Figure 7b shows the effect of DNN adversarial samples on attacking the HDC classifier. The results are reported for the MNIST dataset. Our evaluation shows that DNN adversarial samples can fool the

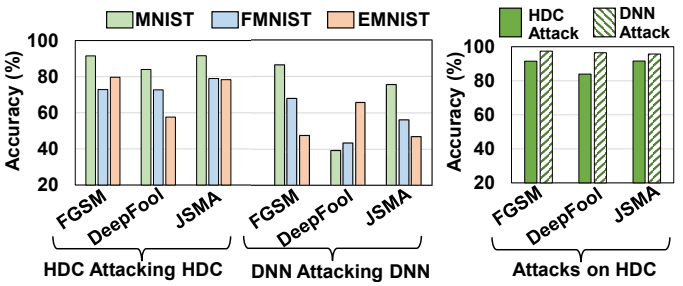

Fig. 7. (a) Adversarial accuracy of HDC and DNN models for different datasets. (b) adversarial accuracy of HDC attacked by HDC and DNN adversarial samples on MNIST dataset.

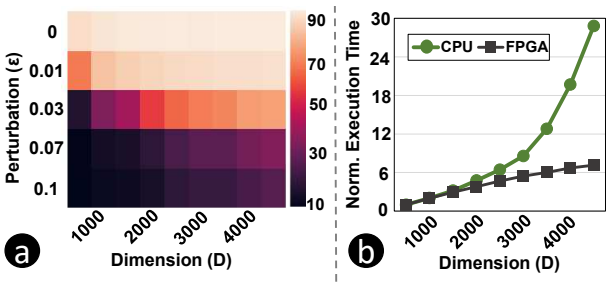

Fig. 8. (a) HDC accuracy to adversarial samples using different epsilons and dimensions, (b) HDC execution time running on CPU and FPGA.

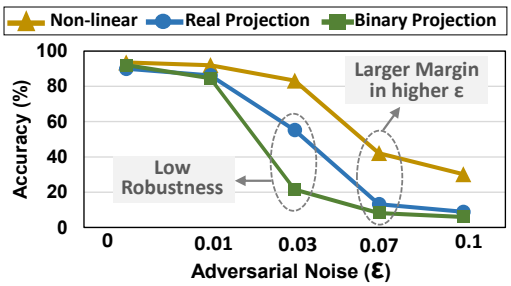

Fig. 9. Impact of encoding method on HDC robustness to adversarial attacks.

HDC model. However, these samples are less effective than HDC adversarial samples in attacking our model. This evaluation indicates: (1) the effectiveness of the samples generated by HDC for attacking HDC models. In particular, it validates our framework's effectiveness in generating gradient-based attacks. (2) The internal robustness of HDC models to samples generated with non-HDC-based approaches. Our results show that the attacks made by DNN models using FGSM (DeepFool and JSMA) have a 5.8% (12.5% and 4.1%) lower success rate on the HDC model than attacks made by HDC adversarial samples.

E. Dimensionality and Adversarial Robustness

The HDC model's dimensionality directly impacts its robustness to adversarial attacks. Figure 8a shows the impact of the hypervector dimensionality on the HDC classification accuracy using the FGSM attack. The results are reported for dimensions varying from $D = 500$ to $D = 4500$. Regardless of the epsilon value, an HDC model with higher dimensionality provides more robustness to adversarial attacks. For example, increasing dimensionality from $D = 500$ to $D = 4500$ improves the HDC classification accuracy by 2.56%. Even from the smallest $\epsilon = 0.01$, the classification accuracy increases up to 30.77%. Our evaluation shows that the dimensionality of the model is a key aspect of the robustness of HDC models, especially for adversarial noise of large magnitude. Note that we can use higher dimensionality to enhance HDC robustness to adversarial attacks further. However, that comes at the overhead of higher computational resources required to train and perform testing.

Figure 8b shows the trade-off between the training efficiency and adversarial robustness for FGSM on CPU and FPGA platforms. HDC naturally has parallelism making it suitable for parallel processors. Since CPUs often have a limited number of cores, their computational performance increases linearly with dimensionality. In contrast, FPGAs can provide high computational parallelism as well as pipeline mechanism to accelerate HDC models with higher dimensions. Our evaluation shows that our model can provide higher adversarial robustness with low-performance overhead on custom and parallel platforms.

F. HDC Robustness with Encoding

Here, we evaluate the impact of HDC encoding on the robustness of our model to adversarial attacks. We evaluate the robustness of three HDC encoding methods: (1) our non-linear encoder with periodic activation function, introduced in Section III-A, and (2) state-of-the-art random projection encoding using real values, where the encoding does not use any activation function [4], and (3) binary random projection encoding [47]. Figure 9 shows the effectiveness of all three encoding methods using the same dimensionality. The results are reported for different adversarial epsilons. Our results indicate that our non-linear encoder has significantly higher robustness to malicious attacks. For example, using $\epsilon = 0.1$, the real and binary projection encoding provides 21.34% and 24.24% lower accuracy than our non-linear encoding method. As we explained, this robustness comes from the periodic activation function that makes gradient-based attacks less successful. In addition, we observe that binary projection encoding has the most vulnerability to attacks. These results guide us towards designing more dynamic encoding methods with higher non-linearity that could provide higher accuracy and robustness to gradient-based malicious attacks.

G. HDC Defense Methods

As explained in Section V, we use two defense mechanisms to further enhance the robustness of the HDC model to adversarial samples: data filtering with average pooling and PGD-based adversarial training. We analyze the defense mechanisms for all three attacks. We report the results for adversarial examples generated using the HDC model. Figure 10 shows the accuracy enhancement of the two defense mechanisms. Our results indicate that data filtering improves HDC adversarial accuracy regardless of the attack mechanism. For example, for FGSM, the accuracy enhancements are 10.4%, 6.3%, and 12.9% for MNIST, FMNIST, and EMNIST, respectively. On the other hand, PGD training is also a successful method in improving the robustness of the HDC model to adversarial samples. However, the PGD training impacts HDC accuracy enhancement less than data filtering. In particular, we observe that PGD training failed to improve robustness against JMSA.

VII. Conclusion

In this paper, we introduce a comprehensive study on the robustness of HDC models to perturbations, either on the model or on the inputs. We design mechanisms to protect HDC models against adversarial

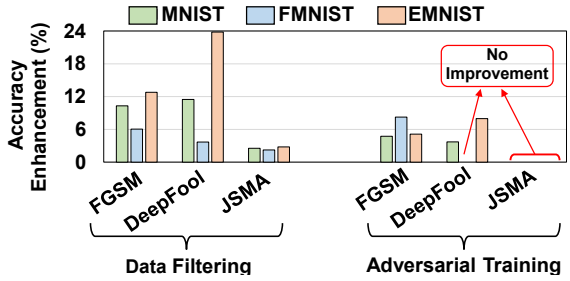

Fig. 10. HDC accuracy enhancement with data filtering & adversarial training.

samples, including data pre-processing and adversarial training. Our evaluation shows that HDC with a proper neural encoding module provides significantly higher robustness to malicious attacks than existing DNN models. In addition, the HDC model has high robustness to adversarial samples generated by DNN. Our study indicates that our defense mechanisms can further protect HDC models and move this technology toward safety-critical applications.

VIII. Acknowledgements

This work was supported in part by DARPA Young Faculty Award, National Science Foundation #2127780, #2319198, #2321840 and #2312517, Semiconductor Research Corporation (SRC), Office of Naval Research, grants #N00014-21-1-2225 and #N00014-22-1-2067, the Air Force Office of Scientific Research under award #FA9550-22-1-0253, and generous gifts from Xilinx and Cisco.

References

[1] A. Krizhevsky *et al.*, "Imagenet classification with deep convolutional neural networks," in *NIPS*, pp. 1097–1105, 2012.
[2] C. Szegedy *et al.*, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.
[3] P. McDaniel, N. Papernot, and Z. B. Celik, "Machine learning in adversarial settings," *IEEE Security & Privacy*, vol. 14, no. 3, pp. 68–72, 2016.
[4] M. Imani *et al.*, "A framework for collaborative learning in secure high-dimensional space," in *CLOUD*, pp. 435–446, IEEE, 2019.
[5] S. Yi *et al.*, "Fog computing: Platform and applications," in *HotWeb*, pp. 73–78, IEEE, 2015.
[6] A. Hernandez-Cane *et al.*, "Onlinehd: Robust, efficient, and single-pass online learning using hyperdimensional system," in *DATE*, pp. 56–61, IEEE, 2021.
[7] P. Poduval *et al.*, "Graphd: Graph-based hyperdimensional memorization for brain-like cognitive learning," *Frontiers in Neuroscience*, p. 5, 2022.
[8] Z. Zou *et al.*, "Biohd: an efficient genome sequence search platform using hyperdimensional memorization," in *Proceedings of the 49th Annual International Symposium on Computer Architecture*, pp. 656–669, 2022.
[9] Z. Zou *et al.*, "Eventhd: Robust and efficient hyperdimensional learning with neuromorphic sensor," *Frontiers in Neuroscience*, vol. 16, 2022.
[10] A. Kazemi, F. Müller, M. M. Sharifi, H. Errahmouni, G. Gerlach, T. Kämpfe, M. Imani, X. S. Hu, and M. Niemier, "Achieving software-equivalent accuracy for hyperdimensional computing with ferroelectric-based in-memory computing," *Scientific Reports*, vol. 12, p. 19201, Nov 2022.
[11] H. E. Barkam *et al.*, "Hdgim: Hyperdimensional genome sequence matching on unreliable highly scaled fefet," in *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6, 2023.
[12] H. Amrouch, M. Imani, X. Jiao, Y. Aloimonos, C. Fermuller, D. Yuan, D. Ma, H. E. Barkam, P. R. Genssler, and P. Sutor, "Brain-inspired hyperdimensional computing for ultra-efficient edge ai," in *2022 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pp. 25–34, 2022.
[13] P. R. Genssler, H. E. Barkam, K. Pandaram, M. Imani, and H. Amrouch, "Modeling and predicting transistor aging under workload dependency using machine learning," *IEEE Transactions on Circuits and Systems I: Regular Papers*, pp. 1–13, 2023.
[14] 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.
[15] A. Rahimi *et al.*, "A robust and energy-efficient classifier using brain-inspired hyperdimensional computing," in *ISLPED*, pp. 64–69, ACM, 2016.
[16] Y. Ni, M. Issa, D. Abraham, M. Imani, X. Yin, and M. Imani, "Hdpg: hyperdimensional policy-based reinforcement learning for continuous control," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pp. 1141–1146, 2022.
[17] Y. Ni *et al.*, "Neurally-inspired hyperdimensional classification for efficient and robust biosignal processing," in *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, pp. 1–9, 2022.
[18] Y. Ni *et al.*, "Algorithm-hardware co-design for efficient brain-inspired hyperdimensional learning on edge," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 292–297, 2022.
[19] H. Chen *et al.*, "Full stack parallel online hyperdimensional regression on fpga," in *IEEE ICCD 2022*, pp. 517–524, IEEE, 2022.
[20] H. Chen *et al.*, "Darl: Distributed reconfigurable accelerator for hyperdimensional reinforcement learning," in *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, pp. 1–9, 2022.
[21] G. Karunaratne *et al.*, "Robust high-dimensional memory-augmented neural networks," *Nat Commun*, vol. 12, p. 2468, 04 2021.
[22] D. Ma, J. Guo, Y. Jiang, and X. Jiao, "Hdtest: Differential fuzz testing of brain-inspired hyperdimensional computing," *arXiv preprint arXiv:2103.08668*, 2021.
[23] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2015.
[24] R. Wiyatno and A. Xu, "Maximal jacobian-based saliency map attack," 2018.
[25] S.-M. Moosavi-Dezfooli *et al.*, "Deepfool: a simple and accurate method to fool deep neural networks," in *CVPR*, pp. 2574–2582, 2016.
[26] B. Babadi and H. Sompolinsky, "Sparseness and expansion in sensory representations," *Neuron*, vol. 83, no. 5, pp. 1213–1226, 2014.
[27] Z. Zou *et al.*, "Scalable edge-based hyperdimensional learning system with brain-like neural adaptation," in *SC*, pp. 1–15, 2021.
[28] F. Yang and S. Ren, "Adversarial attacks on brain-inspired hyperdimensional computing-based classifiers," 2020.
[29] O. Gungor *et al.*, "Res-hd: Resilient intelligent fault diagnosis against adversarial attacks using hyper-dimensional computing," *arXiv preprint arXiv:2203.08148*, 2022.
[30] W. Chen *et al.*, "Adversarial attacks on voice recognition based on hyper dimensional computing," *JSPS*, 2021.
[31] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2014.
[32] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," 2016.
[33] J. X. Morris *et al.*, "Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp," 2020.
[34] D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits, "Is bert really robust? a strong baseline for natural language attack on text classification and entailment," 2020.
[35] F. Khalid, M. A. Hanif, S. Rehman, J. Qadir, and M. Shafique, "Fademl: Understanding the impact of pre-processing noise filtering on adversarial machine learning," 2018.
[36] M. Imani *et al.*, "A binary learning framework for hyperdimensional computing," in *DATE*, pp. 126–131, IEEE, 2019.
[37] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Advances in neural information processing systems*, pp. 1177–1184, 2008.
[38] B. Schölkopf, "The kernel trick for distances," in *NIPS*, pp. 301–307, 2001.
[39] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "Adversarial attacks and defences: A survey," 2018.
[40] M. Qiu and H. Qiu, "Review on image processing based adversarial example defenses in computer vision," in *IDS*, pp. 94–99, 2020.
[41] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2017.
[42] L. Fan, F. Zhang, H. Fan, and C. Zhang, "Brief review of image denoising techniques," *Visual Computing for Industry, Biomedicine, and Art*, vol. 2, p. 7, Jul 2019.
[43] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.
[44] G. Cohen *et al.*, "Emnist: an extension of mnist to handwritten letters," 2017.
[45] H. Xiao *et al.*, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," 2017.
[46] M. Imani, Y. Kim, T. Worley, S. Gupta, and T. Rosing, "Hdcluster: An accurate clustering using brain-inspired high-dimensional computing," in *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1591–1594, 2019.
[47] M. Imani *et al.*, "Bric: Locality-based encoding for energy-efficient brain-inspired hyperdimensional computing," in *IEEE/ACM DAC*, pp. 1–6, 2019.