PoisonHD: Poison Attack on Brain-Inspired Hyperdimensional Computing

Ruixuan Wang, Xun Jiao
ECE Department, Villanova University
{rwang8, xun.jiao}@villanova.edu

***Abstract*—While machine learning (ML) methods especially deep neural networks (DNNs) promise enormous societal and economic benefits, their deployments present daunting challenges due to intensive computational demands and high storage requirements. Brain-inspired hyperdimensional computing (HDC) has recently been introduced as an alternative computational model that mimics the "human brain" at the functionality level. HDC has already demonstrated promising accuracy and efficiency in multiple application domains including healthcare and robotics. However, the robustness and security aspects of HDC has not been systematically investigated and sufficiently examined. Poison attack is a commonly-seen attack on various ML models including DNNs. It injects noises to labels of training data to introduce classification error of ML models. This paper presents `PoisonHD`, an HDC-specific poison attack framework that maximizes its effectiveness in degrading the classification accuracy by leveraging the internal structural information of HDC models. By applying `PoisonHD` on three datasets, we show that `PoisonHD` can cause significantly greater accuracy drop on HDC model than a random label-flipping approach. We further develop a defense mechanism by designing an HDC-based data sanitization that can significantly recover the accuracy loss caused by poison attack. To the best of our knowledge, this is the first paper that studies the poison attack on HDC models.**

I. Introduction

Artificial intelligence (AI), especially machine learning (ML) paradigms, promises enormous societal and economic success. Learning methods such as deep neural networks (DNNs) require a large amount of training data and also extensive labor for data labelling for the purpose of supervised learning. However, the environment of the ML model deployed, as well as the source of data, are not always guaranteed and secure [24]. It is an accepted fact that an adversary is able to inject bad or false data to the model training pool. As a result, the model learns false information and the decision boundary will get shifted in a certain way. This attack is known as the data poisoning attack [24]. Various studies have demonstrated that poison attacks can largely compromise the performance of learning models [24], [11]. To mitigate the influence of poison attacks, some successful defenses employ anomaly detection-based approach based on a verified subset of the training set to exclude the maliciously injected data [18], [19].

Recently, as an emerging alternative computing paradigm to DNNs, the brain-inspired hyperdimensional computing (HDC) has demonstrated promising capability in a wide range of applications including computer vision [6], [16], audio [8], robotics [14], and natural language processing [21]. HDC is inspired by the working mechanism of human brain which computes embedded and abstract patterns of neural signals rather than actual numbers [10]. This inspires the formulation of hyperdimensional vectors with fully distributed holographic representation and (pseudo) randomness. Compared to DNNs, HDC has shown advantages such as smaller model size, less computation cost, and one-shot learning capability, making it a promising alternative in low-cost computing platforms [5].

Despite the intensive studies on applying HDC for supervised learning tasks (i.e., classification) in various problem domains [5], there is no known study on poison attack of HDC. Although the influence of data poisoning attack has been well studied in DNN models [24], these existing methods are not applicable to HDC models because unlike traditional ML methods such as DNNs with a well-defined differentiable mathematical architecture, HDC architecture is not differentiable and is less application-agnostic. The encoding of HDC is largely unique for different applications and relies on random indexing to project data onto vectors in a hyperdimensional space [21], adding difficulty to efficiently acquire adequate information to guide the poison process. This paper develops an poison attack framework specifically designed for HDC models and analyze the robustness of the HDC model under poison attack.

This paper makes the following technical contributions:

- To the best of our knowledge, this paper presents the first study on poison attack of HDC models and show that HDC is indeed vulnerable to poison attack. We present **`PoisonHD`** that can automatically generate poison attacks for a given HDC model by injecting false data with flipped labels.
- We develop a confidence-based label-flipping method specifically for HDC models which can maximize the effects of poison attacks within a certain label-flipping budget. By comparing **`PoisonHD`** with a random label-flipping approach on three datasets, we show that **`PoisonHD`** consistently results in higher accuracy loss to HDC models.
- We develop a HDC-specific data sanitization approach to defend HDC models against poison attack. Results on three datasets show that the defense mechanism can significantly recover the accuracy loss of HDC models under poison attack.

II. Related Work

HDC, also known as vector-symbolic architecture (VSA), was first proposed by P. Kanerva [10]. Currently, HDC research largely focuses on two aspects: applications of HDC to different domains, and hardware acceleration of HDC. The application scenarios of HDC expands from natural language processing [10], [21] to biomedical engineering [20], [15], robotics [14], and computer vision [16]. HDC is used in modern robotics to perform active perception by integrating the sensory perceptions experienced by an agent with its motoric capabilities, which is vital to autonomous learning agents [14]. HDC is also used in the computer vision area; [16] proposed a framework based on HDC to concrete local descriptors in the image and improved the performance in place recognition tasks. Another research focus is to accelerate HDC processing on hardware. Computation reuse is leveraged in FPGA to achieve energy-efficient HDC processing [7] . *HDC-IM* [12] 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 [23] and 3D IC [26].

On the other hand, data poisoning turns into a serious security issue with the development of ML. One common indication of data poisoning is data poisoning attack. Data poisoning attack has been classified to “targeted attack” and “indiscriminate attack” based on the aim of the attacker. According to [1], the data poisoning attack can be “exploratory”, which means that the attacker discovers information of the classifier [4], or “causative” denoted to influence the performance of the classifier [13]. [17], [25] also proved that data poisoning exists in online learning and federated learning. Despite the growing popularity of HDC, there is no known study on poison attack on HDC models.

This paper presents the first effort in such a paradigm by demonstrating and developing HDC-specific poison attacks. Further, a defense mechanism is developed to mitigate data poisoning attacks on HDC.

III. HDC Background

A. HDC Basics

The basic component of HDC is called hypervectors (HVs). HVs are high-dimensional (dimension $D = 10,000$), holographic vectors with random and identically distributed (i.i.d.) elements [10]. An HV with d dimensions can be denoted as $\overrightarrow{H} = \langle h_1, h_2, \ldots, h_d \rangle$, where h_i represents the i element of the HV. HVs are used to accommodate and represent information in different scales and levels. When the dimensionality is sufficiently high, e.g., $D = 10,000$, any two random HVs are nearly orthogonal [10]. HDC utilizes this quasi-orthogonality property and different operations to support as means of producing aggregations of information or creating representations of new information.

HDC provides three basic operations, addition ($+$), multiplication ($*$) and permutation (ρ), as elaborated in Eq. 1. Addition and multiplication take two input HVs as operands and perform element-wise add or multiply operations on the two HVs. Permutation takes one HV as the input operand and performs circular shifting. Each of the operations have no modification on the dimensionality of the input HVs, i.e. the input and the output HVs are in the same dimension.

$$\begin{aligned}
&\overrightarrow{H_x} + \overrightarrow{H_y} = \langle h_{x1} + h_{y1}, \ldots, h_{xd} + h_{yd} \rangle \\
&\overrightarrow{H_x} * \overrightarrow{H_y} = \langle h_{x1} * h_{y1}, \ldots, h_{xd} * h_{yd} \rangle \\
&\rho^1(\overrightarrow{H}) = \langle h_d, h_1, \ldots, h_{d-1} \rangle
\end{aligned} \tag{1}$$

Similarity Measurement In HDC, we use the similarity metric δ to measure the similarity between two HVs, as well as the similarity of information represented by HV. For quantifying the similarity, we use cosine similarity in this paper. A higher similarity δ between two HVs shows that these two HVs have more information in common, or vice versa.

In this experiment, we employ bipolar HVs, which means elements in an HV are either -1 or 1. In this end, the similarity metric δ denotes as Eq. 2. When the dimensionality is sufficiently high (e.g., $D = 10,000$), any two random bipolar HVs are nearly orthogonal [10].

$$\delta(\overrightarrow{H_x}, \overrightarrow{H_y}) = \frac{\overrightarrow{H_x} \cdot \overrightarrow{H_y}}{||\overrightarrow{H_x}|| \times ||\overrightarrow{H_y}||} \tag{2}$$

B. HDC Process

To develop HDC models for classification on specific tasks, we need three main processes including **Encoding**, **Training**, and **Inference**. The detailed processes are discussed in the following of this section.

Encoding is the process to project a real-world feature vector into an HV. In our paper, we follow a shifting-based encoding strategy [9]. In general, for each feature vector $\overrightarrow{F_n} = \langle f_1, f_2, \ldots, f_n \rangle$, every feature value in the $\overrightarrow{F_n}$ is not integer. In this domain, for each feature value f_i, the encoding approach requests a set of values between the minimum and maximum value of the feature. In our experiment, we first generate N seed HVs in item memory which are correlated to N quantization level, and represent each interval by a HV. Then we discretize the real number feature value and index the corresponding seed HV. For instance, if a feature value f_i falls into the k^{th} interval among the N quantization level, then its corresponding seed HV is the k^{th} of the N seed HVs. Then, we employ the permutation operation to embed the information of feature position into the related seed HV. As permutation operation reflects the spatial change of information, we include the information of feature position by deploying a circular rotation on each seed HV. At the end of encoding progress, we aggregate all seed HVs corresponding to all feature values into one HV $\overrightarrow{H_{F_n}}$ representing the entire feature vector $\overrightarrow{F_n}$. The encoding process as shown in Eq. 3.

$$\overrightarrow{H_{F_n}} = \rho^0(\overrightarrow{s_{f_1}}) + \rho^1(\overrightarrow{s_{f_2}}) + \cdots + \rho^{n-1}(\overrightarrow{s_{f_n}}) \tag{3}$$

Training is the process of establishing the associative memory (AM) over the entire training set. In a binary classification, AM consists of the class HV representing positive (pos) class and negative (neg) class. Eq. 4 illustrates the process of HDC training by adding (bundling) every HV representing every

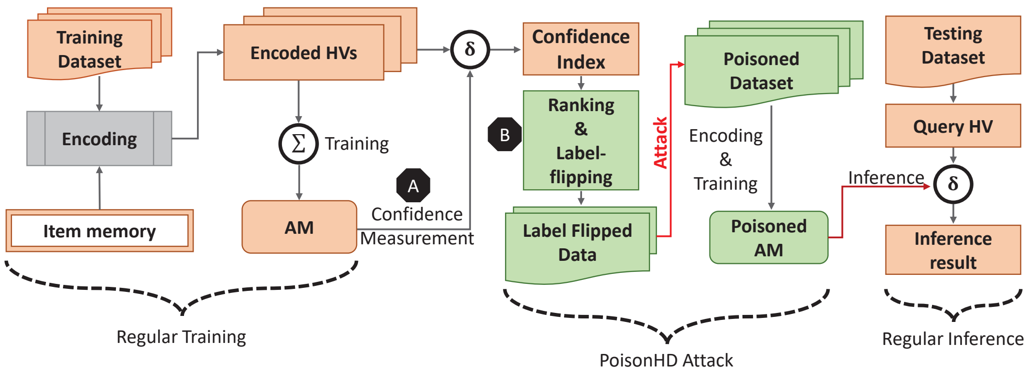


Fig. 1. **PoisonHD** framework.

feature vector sharing the same label. The learning rate lr is a configurable hyperparameter which is fixed for all HVs. After the aggregation, the two HVs in AM represent the information of all positive and all negative sample HVs respectively.

$$\overrightarrow{H_{pos}} = \sum lr * \overrightarrow{H_p} \ , \\ \overrightarrow{H_{neg}} = \sum lr * \overrightarrow{H_n} \tag{4}$$

Inference is the process of using unseen data from the testing set to evaluate the performance of a well-trained HDC classifier. Before making decision, we encode the testing sample q into an HV called query HV $\overrightarrow{H_q}$ following the same encoding process in Eq. 3 based on the same item memory. Then we calculate the cosine distance (δ) between a query HV $\overrightarrow{HV_q}$ and each class HV in the AM to obtain the similarity. As Eq. 5 denotes, the HV in AM with the highest similarity indicated the category of the query HV x.

$$x = argmax(\delta(\overrightarrow{H_q}, \overrightarrow{H}_{pos}), \delta(\overrightarrow{H_q}, \overrightarrow{H}_{neg})) \tag{5}$$

IV. PoisonHD FRAMEWORK

Threat Model: We consider the following assumption for the data poisoning attack: the attacker has full knowledge of the HDC algorithm (i.e., encoding, training and similarity/distance measurement of a given testing sample). The attacker is allowed to add extra modified samples to the datasets but has no permission to tamper existing data and corresponding labels, which is following the causative attack model. This assumption is consistent with existing studies such as [24].

Overview: Typically, the most straightforward data poisoning attack is via a random label-flipping [22]. However, a random injection is amorphous and unguided, thus may not generate effective false information that can introduce strong performance loss. In this paper, we propose **PoisonHD**, a confidence-based label-flipping method specifically designed for poisoning HDC models. The method is divided into two phases as follows.

A. *Confidence Measuring*

As the module (A) shows in Fig. 1, the **PoisonHD** have full access to the HDC model, including encoded training HVs as well as the well-established AM. In this paper we define the confidence of each HV, which is a measurement of similarity deviation between different class HVs and each HV, shown in Fig. 6. For example, we measure the similarity of each training HV HV_i with the positive and negative class HV separately, as denoted in Eq. 6. For example, if the cosine similarity between HV_i and HV_{pos} equals d_{i1} and the cosine similarity between HV_i and HV_{neg} equals d_{i2}, the confidence $Confidence_i$ can be determined by $|d_{i1} - d_{i2}|$. The confidence $Confidence_i$ reflects the discernment of the HDC model on the data sample corresponding to HV_i.

$$Confidence_i = |\delta(H_i, H_{pos}) - \delta(H_i, HVneg)| \tag{6}$$

B. *Confidence based Ranking and Label-flipping*

According to the strategy in **PoisonHD**, after we calculate the confidence of all the training HVs using Eq. 6, we rank all the encoded training data, by its confidence in a descending order. Based on some previous research, an arbitrary label-flipping will cause a negative influence on a Machine learning model [2]. While in the data poisoning domain the purpose of the attacker is to maximally fail the HDC model and reduce the performance of such model. To this end, the top-N highest confidence training samples can be considered as the most vulnerable samples in the training set against label-flipping. When **PoisonHD** poison HDC model on these samples, we can achieve the maximum performance drop on the selected HDC model.

According to our approach, we rank all the training HVs by its confidence. According to our approach, we rank all the training HVs by its confidence. In this ranking, higher confidence indicates the information in the HV_i is more similar to the related class HV than to another class HV. Meanwhile, when we apply label-flipping on a high confidence sample, the negative influence caused by the mismatch between the content and the label of samples is relatively higher than the negative influence of low confidence samples. Therefore

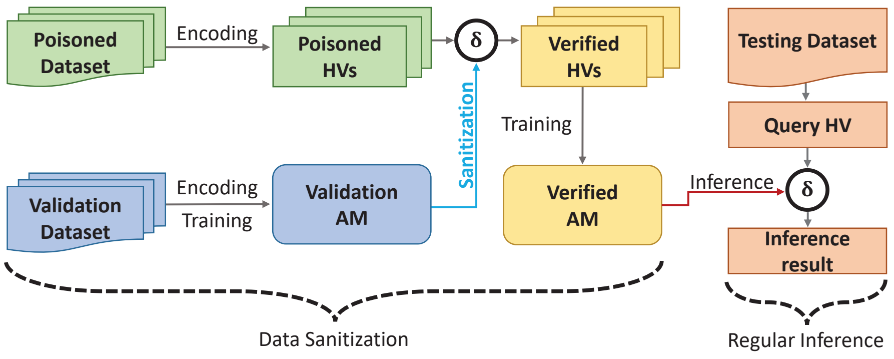


Fig. 2. HDC-specific data sanitization defense.

PoisonHD index the top-N highest confidence samples as the target samples of poisoning attack. To this end, if we deploy label-flipping on the top-N training HVs ranked by confidence, the poison attack will cause the maximum adverse impact on AM.

According to the earlier threat model of data poisoning attack, the existing training data in the training dataset cannot be modified but the attackers are allowed to add new data samples into the training pool. To this end, **PoisonHD** adds the unchanged data samples with flipped labels into the original dataset, which means the dataset contains two identical samples but with opposite label [3]. When **PoisonHD** attaches the label-flipped samples into the training dataset and this dataset is considered as a poisoned dataset. After that, every HDC model follows the normal HDC training process, using the same item memory and encoding method, and establishes an AM (which is poisoned) over the poisoned dataset.

V. Defense Against PoisonHD

To defend the data poisoning attack, we propose an HDC-specific data sanitization solution. The data sanitization works directly on the poisoned dataset which contains label flipped samples. The objective of data sanitization is to automatically remove the suspicious samples which may get poisoned.

In our experiment, we develop an HDC-based data sanitization approach, which is also known as an "Oracle defense" mentioned in [24]. For deploying such defense method, we require an extra verified dataset and this verified dataset is not accessible to the attacker. The relationship between data samples and data labels in the extra dataset are validated and correct. In this paper we propose an HDC based data sanitization approach which is illustrated in Fig. 2. The basic idea is to train an outlier detector using the verified dataset and filter the ambiguous samples. First of all, we follow the regular encoding and training process to establish an HDC model AM_{val} based on the verified dataset, where AM_{val} is our outlier detector. Then we deploy an HDC based data sanitization defense approach to filter out the poisoned samples in the poisoned training set by checking the relationship between the data samples and data labels. In our approach the performance of data sanitization defense only depends on the number of verified clean training samples. The detailed algorithm of HDC based data sanitization defense is described as Algorithm.1. In this algorithm first we encode all the poisoned samples into poisoned HVs with the same encoding method (we assume the dataset is poisoned). Then we employ data sanitizer for the defense against the data poisoning attack.

Algorithm 1 Data Sanitization Defense

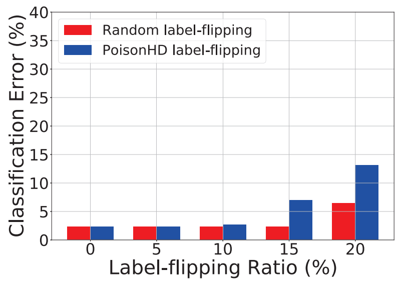
```
Input Poisoned HV $T = \{HV_1, ..., HV_N\}$; Validation AM $AM_{val} = \{HV_{pos'}, HV_{neg'}\}$
Output Verified HV Index Array $C = [c_1, ..., c_m]$
1: $C = \{\}$
2: for $HV_i$ in $T$ do \* Perform data sanitization */
3:     $label_{pred} \leftarrow argmax(\delta(HV_{pos'}, HV_i),\ \delta(HV_{neg'}, HV_i))$
4:     if $label_{pred} = label_{true}$ then
5:         $C.append(i)$
6:     end if
7: end for
8: return $C$
```

As Algorithm. 1 presents, we employ the validation AM AM_{val}, including two class HVs $HV_{pos'}$ and $HV_{neg'}$, and traverse all the poisoned HVs to examine if the label is matching with the content. For example, if the label assigned by the validation AM matches the true label in the dataset, we save such HV as a verified HV. The verified HV is available to build a normal AM. On the other hand, if the label assigned by the data sanitizer is different from the true label, for example, a "positive" label assigned by data sanitizer while the true label is "negative", or vice versa. In this scenario the HV is considered as a noxious HV and will not count in the set of verified HVs. After a one-time sanitization, the verified HVs can be used to normally train an HDC model and test on query HVs following the regular process.

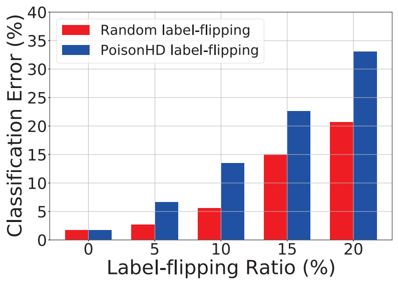
VI. Experimental Results

A. Experiment Setup

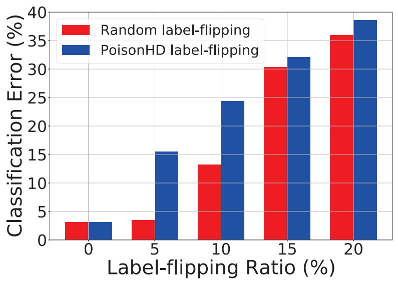
We deploy **PoisonHD** on three different datasets, **MNIST-1-7** [3], **Dog Fish** [11] and **Breast Cancer** [19]. These three datasets focus on real world binary classification tasks and are ordinary benchmarks for the classification task. For **MNIST-1-7** dataset, the classification task is to distinguish hand-written digits



(a) MNIST-1-7

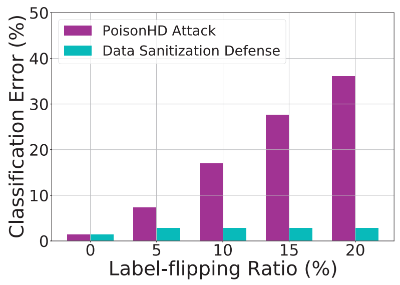


(b) Dog Fish

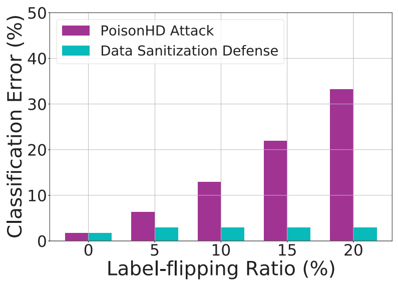


(c) Breast Cancer

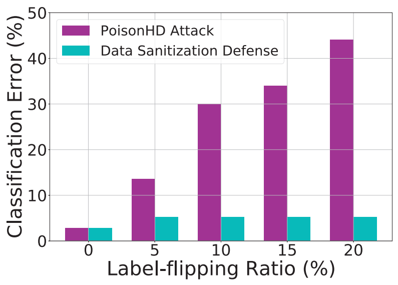
Fig. 3. Classification error of 10000-dimension HDC models under different label-flipping ratio.



(a) MNIST-1-7



(b) Dog Fish



(c) Breast Cancer

Fig. 4. Classification error of 10000-dimension HDC models under **`PoisonHD`** attack and data sanitization defense.

between "1" and "7"; there are 13007 different images with $28 * 28$ resolution. For **`Dog Fish`** dataset, we use the Inception-v3 features [11] of 1800 different images of dogs and fishes, where each feature vector is of 2048 length. **`Breast Cancer`** is a medical dataset containing 569 diagnostic results of breast cancer that is either malignant or benign based on 30 features. There is no extra pre-processing except that we normalize the training dataset to the range $[-1, 1]$. For each dataset, we divide it into three parts, a training set (70%), which is the potential target of data poisoning attack, a testing set (20%), and a validation set (10%).

We first train the HDC model using the original (clean) training set to set a baseline model. The baseline model is also the target of data poisoning attack in **`PoisonHD`**. Then we use **`PoisonHD`** to implement data poisoning attack into the training set and examine the degradation of classifier performance using the same testing set. For each experiment, we repeat 5 times and calculate the average accuracy as the presented results. The baseline HDC accuracy on **`MNIST-1-7`**, **`Dog Fish`**, **`Breast Cancer`** are 98.6%, 98.3% and 96.9% respectively.

B. *Effects of **`PoisonHD`** attack*

In the first experiment we present the influence of different ratios of data poisoning on the three datasets. Without loss of generality, we inject different label-flipping ratios ranging from 5% to 20%. It is worth mentioning that the data poisoning attack only targets the training dataset, which is 70% of the full data as mentioned earlier. The blue bars are the classification error (accuracy drop) of the HDC model under a specific ratio of poisoning samples in the training set. Without loss of generality, we repeat 5 independent experiments and calculate the average classification error. In the meantime, to avoid over-fitting, we lessen the learning rate of HDC training on **`MNIST-1-7`** dataset as $lr = 0.1$, and $lr = 1$ on the **`Dog Fish`** and **`Breast Cancer`** dataset.

According to the classification error illustrated in Fig. 3, our experiment points out the weakness of HDC when against data poisoning attacks. Based on our experiment result, we notice that HDC models are severely damaged under data poisoning attacks, even under a random flipping where we randomly inject label-flipping on a certain ratio of samples. For instance, when we inject label-flipping on 15% training samples, the classification error of random label-flipping attack are 3.3%, 14.9% and 30.3% on each dataset respectively. In the meantime, the classification errors under **`PoisonHD`** label-flipping attack are 7.1%, 22.6% and 32.2%.

One observation is that the classification error under random label-flipping attack is always lower than the **`PoisonHD`** based label-flipping attack over every poisoning ratio on all the three datasets, which demonstrates the feasibility and effectiveness of **`PoisonHD`**. Moreover, the classification error caused by **`PoisonHD`** can be considered as the "upper limit" error since **`PoisonHD`** based poisoning attack leads to the maximum level of negative influence on a specific HDC model on the basis of our experimental result.

C. *Effects of Data Sanitization*

In our second experiment, after we use **`PoisonHD`** to poison the HDC model based on a certain ratio of poisoning samples, we employ the HDC-specific data sanitization defense approach to mitigate the negative influence caused by data poisoning attack. The classification error under different ratios of data poisoning attack is denoted as the purple bars in Fig.4: Compared with the baseline classification error the HDC model of three datasets suffer from 26%, 20% and 31% accuracy drop because of 15% data poisoning attack, after the data sanitization defense, the classification error of the HDC model leads to only 1.4%, 1.2% and 2.5% decrease from the baseline, as the cyan bars show in Fig.4. The similar sanitization performance is even achieved under a 20% data poisoning attack. This indicates that, while the validation dataset only has 10% of training samples, we can still recover a poisoned HDC model to a workable classifier with acceptable classification accuracy based on our data sanitization defense approach.

VII. Conclusion

For the first time, this paper investigates the effects of data poisoning attack on brain-inspired hyperdimensional computing algorithms. Our experiments demonstrate that HDC is vulnerable to data poisoning, especially label-flipping attacks. By leveraging the inherent structural information of HDC processing, we propose **PoisonHD**, an HDC-specific data poisoning attack framework that maximizes its effectiveness in degrading the classification accuracy. **PoisonHD** uses a confidence-based measurement to identify the critical labels to be flipped that can introduce most significant effects to HDC performance. We further propose a HDC-specific data sanitization defense to recover the effects of data poisoning attack. Experimental results show that we are able to significantly recover the accuracy loss after data sanitization. Our future work will consider label-flipping attack in multi-classification tasks, which is a more sophisticated problem. Moreover, we will consider a diverse set of attack types such as backdoor attack, shilling attack, and model stealing attack that can be launched on HDC models, and develop corresponding defense mechanisms.

References

[1] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D Joseph, and J Doug Tygar. Can machine learning be secure? In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 16–25, 2006.

[2] Battista Biggio, Blaine Nelson, and Pavel Laskov. Support vector machines under adversarial label noise. In *Asian conference on machine learning*, pages 97–112. PMLR, 2011.

[3] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*, 2012.

[4] Prahlad Fogla, Monirul I Sharif, Roberto Perdisci, Oleg M Kolesnikov, and Wenke Lee. Polymorphic blending attacks. In *USENIX security symposium*, pages 241–256, 2006.

[5] Lulu Ge and Keshab K Parhi. Classification using hyperdimensional computing: A review. *IEEE Circuits and Systems Magazine*, 20(2):30–47, 2020.

[6] Michael Hersche, Edoardo Mello Rella, Alfio Di Mauro, Luca Benini, and Abbas 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*, pages 169–174, 2020.

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

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

[9] Aditya Joshi, Johan T Halseth, and Pentti Kanerva. Language geometry using random indexing. In *International Symposium on Quantum Interaction*, pages 265–274. Springer, 2016.

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

[11] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, pages 1885–1894. PMLR, 2017.

[12] Jialong Liu, Mingyuan Ma, Zhenhua Zhu, Yu Wang, and Huazhong Yang. Hdc-im: Hyperdimensional computing in-memory architecture based on rram. In *2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pages 450–453. IEEE, 2019.

[13] Daniel Lowd and Christopher Meek. Adversarial learning. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 641–647, 2005.

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

[15] Ali Moin et al. An emg gesture recognition system with flexible high-density sensors and brain-inspired high-dimensional classifier. In *ISCAS*, 2018.

[16] Peer Neubert and Stefan Schubert. Hyperdimensional computing as a framework for systematic aggregation of image descriptors, 2021.

[17] Naman Patel, Prashanth Krishnamurthy, Siddharth Garg, and Farshad Khorrami. Bait and switch: Online training data poisoning of autonomous driving systems. *arXiv preprint arXiv:2011.04065*, 2020.

[18] Andrea Paudice, Luis Muñoz-González, Andras Gyorgy, and Emil C Lupu. Detection of adversarial training examples in poisoning attacks through anomaly detection. *arXiv preprint arXiv:1802.03041*, 2018.

[19] Andrea Paudice, Luis Muñoz-González, and Emil C Lupu. Label sanitization against label flipping poisoning attacks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 5–15. Springer, 2018.

[20] Abbas Rahimi, Simone Benatti, Pentti Kanerva, Luca Benini, and Jan M Rabaey. Hyperdimensional biosignal processing: A case study for emg-based hand gesture recognition. In *2016 IEEE International Conference on Rebooting Computing (ICRC)*, pages 1–8. IEEE, 2016.

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

[22] Elan Rosenfeld, Ezra Winston, Pradeep Ravikumar, and Zico Kolter. Certified robustness to label-flipping attacks via randomized smoothing. In *International Conference on Machine Learning*, pages 8230–8241. PMLR, 2020.

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

[24] Jacob Steinhardt, Pang Wei Koh, and Percy Liang. Certified defenses for data poisoning attacks. *arXiv preprint arXiv:1706.03691*, 2017.

[25] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. Data poisoning attacks against federated learning systems. In *European Symposium on Research in Computer Security*, pages 480–501. Springer, 2020.

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