PRID: Model Inversion Privacy Attacks in Hyperdimensional Learning Systems

Alejandro Hernández-Cano*, Rosario Cammarota$^{\psi}$, Mohsen Imani†*
*Universidad Nacional Autónoma de México, $^{\psi}$Intel Labs, †University of California Irvine
*Email: m.imani@uci.edu

***Abstract*—Hyperdimensional Computing (HDC) is introduced as a promising solution for robust and efficient learning on embedded devices with limited resources. Since HDC often runs in a distributed way, edge devices need to share their model with other parties. However, the learned model by itself may expose information of the train data, resulting in a serious privacy concern. This paper is the first effort to show the possibility of a model inversion attack in HDC and provide solutions to overcome the challenges. HDC performs learning tasks after mapping data points into high-dimensional space. We first show the vulnerability of the HDC encoding module by introducing techniques that decode the high-dimensional data back to the original space. Then, we exploit this invertibility to extract the HDC model's information and reconstruct the train data just by accessing the model. To address the privacy challenges, we propose two iterative techniques which scrutinize HDC model from a privacy perspective: (i) intelligent noise injection that identifies and randomizes insignificant features of the model in the original space, and (ii) model quantization that removes model's recoverable information while teaches the model iteratively to compensate the possible quality loss. Our evaluation over a wide range of classification problems indicates that our solution reduces the information leakage by 92% (66%) while having less than 5% (3%) impact on the learning accuracy.**

I. Introduction

Many applications run machine learning algorithms to assimilate the data collected in the swarm of devices on the Internet of Things (IoT). Sending all the data to the cloud for processing is not scalable, cannot guarantee a real-time response, and is often not desirable due to privacy and security concerns [1]–[6]. Much of IoT data processing will need to run at least partly on devices at the edge [7]–[10]. However, the high computational complexity and memory requirement of existing Neural Networks (DNN) hinder usability to a wide variety of real-life embedded applications where the device resources and power budget is limited [11]–[14]. Therefore, we need alternative learning methods to train on the less-powerful IoT devices while ensuring model accuracy and privacy.

Hyperdimensional Computing (HDC) is introduced as a promising solution for robust and efficient learning. HDC is motivated by the understanding that the human brain operates on *high-dimensional* representations of data originated from the large size of brain circuits [15]. It thereby models the human memory using points of a high-dimensional space, that is, with *hypervectors*. HDC performs a learning task after mapping data into high-dimensional space. This encoding is performed using a set of pre-generated *base vectors*. HDC is well suited to address several learning tasks in IoT systems as: (i) HDC is computationally efficient and amenable to hardware level optimization [16]–[18], (ii) it is a computational paradigm that can be applied to a wide range of learning and cognitive problems [19]–[21], and (iii) it provides strong robustness to noise – a key strength for IoT systems [20], [22].

Although HDC enables ultra-efficient learning, the lack of trustworthy learning limits its practical application in real-world IoT systems. Privacy is one of the key challenges of machine learning algorithms as the trained model may expose the information of the training data. In HDC, the privacy is a bigger concern as: (i) HDC has a transparent model and often runs in a distributed system where devices need to share their trained model to other parties, (ii) the HDC encoding module is invertible, meaning that the high-dimensional data can be decoded back to original space just by accessing the base vectors (encoding key). Unfortunately, these bases need to be shared by all parties involved in the training or inference process. Prior work tried to make address trustworthiness challenges in HDC. For example, work in [23] introduced the first adverserial attack on the HDC model. Work in [24] exploited multi-party computation (MPC) to enable secure collaborated learning by assigning a private key (base hypervector) to each user. Work in [25] securitized the HDC from a differential privacy perspective by adding noise to encoded samples during training and inference. However, these approaches assumed the trained model does not expose any information on the enclosed dataset.

This paper is the first effort toward the possible model inversion attack in HDC. We propose a framework that not only checks the existence of an inference data in the train set, but also reconstructs the train data only by accessing the HDC trained model. We also propose solutions to overcome privacy challenges. We summarize our main contributions here:

- We proposed PRID, a novel framework to enhance HDC privacy against model inversion attack.We show the vulnerability of HDC encoding by introducing analytical and learning-based techniques to decode the high-dimensional data back to the original space.
- PRID exploits the invertibility of the encoding module to reconstruct and estimate data points used for model training. PRID first decodes the HDC model back to original space and gets a rough estimation of train data used for model training. For a given query data, PRID reconstructs a train data by identifying query features, e.g., pixels, that are possibly used for model training. Then, PRID reconstructs new data by replacing the non-used query features with features from the decoded model. An iterative process will reconstruct train data that has been used for model learning.
- We propose two iterative techniques to scrutinize HDC model against model inversion attack: (i) *intelligent noise injection* that identifies and randomizes insignificant model features in original space, and (ii) *model quantization* that degrades the HDC decoding capability in the HDC model. PRID integrates both approaches with an iterative learning framework that compensates for the possible quality loss caused by noise injection or quantization.

We evaluate PRID on a wide range of classification problems and observe that PRID almost to eliminate data exposure, e.g., it reduces the information leakage by 92% (66%) while having less than 5% (3%) impact on the learning accuracy. Our code is available open-source[1]

II. Preliminary

A. Hyperdimensional Learning

Hyperdimensional computing (HDC) consists of three modules: encoding, training, and associative search.

978-1-6654-3274-0/21/$31.00 ©2021 IEEE

[1]https://gitlab.com/biaslab/pridhd

Encoding: is the first operation involved in HDC. The goal of the encoding module is to map data in a space that data points can be classified using a simpler classifier. Depending on data type, prior work introduced different encoding modules [24], [26], [27]. Here, we consider the state-of-the-art encoding method for feature vector [24]. Let us consider an encoding function that maps a feature vector $\vec{F} = \{f_1,\ f_2, \ldots,\ f_n\}$, with n features ($f_i \in \mathbb{R}$) to a hypervector $\vec{\mathcal{H}} = \{h_1,\ h_2, \ldots,\ h_D\}$ with D dimensions ($h_i \in \mathbb{R}$). We generate each dimension of encoded data by calculating: $\vec{\mathcal{H}} = \sum_{k=0}^{n-1} f_k \cdot \vec{\mathcal{B}}_k$, where $\vec{\mathcal{B}}_k \in \{-1, +1\}^D$ are randomly generated base hypervectors. Due to nature of random generation, these hypervectors are nearly orthogonal: $\delta(\vec{\mathcal{B}}_{k_1}, \vec{\mathcal{B}}_{k_2}) \simeq 0$, where δ denotes the cosine similarity. Thus, each base hypervector can retain the spatial or temporal location of each feature in an input. Instead of using actual feature values, prior work [26] performed vector quantization to represent feature values using highly correlated hypervectors. However, this comes at the penalty of quality loss. In HDC, the base hypervectors $\{\vec{\mathcal{B}}_1, \vec{\mathcal{B}}_2, \cdots, \vec{\mathcal{B}}_n\}$ are are generated once in offline and then can be used for rest of the classification.

Training: of HDC starts with accumulating all encoded hypervectors corresponding to each class. The result will be k hypervectors with D dimensions, where k is the number of classes. Assuming there are $\mathcal{J}$ inputs having label l: $\vec{\mathcal{C}}_l = \sum_j^{\mathcal{J}} \vec{\mathcal{H}}_j^l$. HDC also supports iterative training [24], but that comes at the cost of higher training time and energy.

Inference: of HDC starts by encoding the test data into high-dimensional space using the same encoding module used for training. The encoded data is called *query hypervector* $\vec{\mathcal{H}}$. Next, we compare the similarity (δ) of $\vec{\mathcal{H}}$ and all class hypervectors to find a class with the highest similarity.

B. Privacy in HDC

The existing HDC learning systems have major privacy and security challenges. Work in [24] introduced SecureHD as a framework for secure collaborative learning in high-dimensional space. SecureHD considers a scenario that cloud is untrusted and then exploits multi-party computation (MPC) to generate a private base hypervector for each client. Work in [23] designed the first adversarial attack for HDC. Work in [25] exploits the well-known noise injection method to enable differential privacy in high-dimensional space. All existing solutions assumed the trained model does not expose any information on the enclosed dataset. In contrast, this paper is the first effort toward securitizing the HDC model against model inversion attack. Since HDC often works in federated learning, users need to share their model with other parties. This can possibly reveal the information of train data points that have been used for model learning. In this work, we show the possibility of a model inversion attack and proposed general solutions to securitize the HDC model.

III. HDC Model Inversion Attack

In HDC, the encoding module is invertible, meaning that the encoded hypervector can be decoded back to the original space. In fact, the HDC encoding module is secure unless one has access to the base hypervectors used for encoding [24]. This invertibility can provide the following advantages: (i) transparent model for reasoning about the decision made by the model, (ii) secure storage to store the encoded data in the cloud [24]. However, this invertibility creates several privacy challenges as it can be used to extract useful information from the model. Model inversion attack is one of the significant privacy challenges in learning systems [28], [29]. This attack extracts the information of the train data by accessing the model.

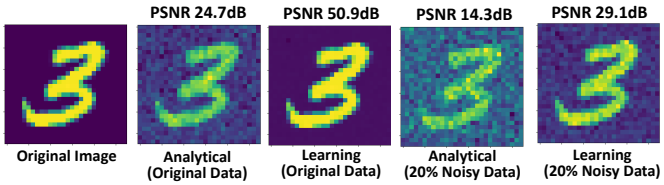


Fig. 1. Data decoding techniques over original and noisy data.

A. Data Recovery in HDC

Previous work has introduced a data recovering method which we refer to as analytical data decoding [24]. Here, we introduce a learning-based method to decode high-dimensional data back to original space.

Analytical Data Decoding: Let us assume $\vec{F}$ is a feature vector which has been encoded to high-dimensional space ($\vec{\mathcal{H}}$) using base hypervectors $\vec{\mathcal{B}}$. PRID decodes each feature in the original space using: $f_i \approx \frac{\vec{\mathcal{B}}_i^T \cdot \vec{\mathcal{H}}}{D}$. The same equation can be used to recover the first estimation of the feature vector: $\vec{F}^{(1)} = \{f_1^{(1)}, f_2^{(1)}, \cdots, f_n^{(1)}\}$. To find better estimation of the feature, we perform an iterative data decoding that reduces the noise of the feature vector. This iterative approach encodes the recovered data back to high-dimensional space ($\vec{\mathcal{H}}^{(1)}$) and subtracts it from the encoded data to create an error hypervector: $\vec{\mathcal{E}}^{t+1} = \vec{\mathcal{H}}^t - \vec{\mathcal{H}}^{t+1}$. The error hypervector will be decoded back to original space ($\vec{E}^{(1)}$) and added by the first feature vector estimation ($\vec{F}^{t+1} = \vec{F}^t + \lambda \vec{E}^t$), where λ is a small constant to ensure convergence. PRID continues this iterative process until generating a feature good estimation of feature vector with less variance.

Learning-based Data Decoding: We also introduce a learning-based approach to extract information from high-dimensional space. We observe the encoding can be reorganized as a matrix multiplication, and thus decoding can be tackled as a linear regression problem, which can be solved with multiple approaches, e.g. using least squares regression. In this experiment, we exploit a neural network structure with a single hidden layer performing a regression task. The network uses n neurons in the hidden layer, which is equal to the number of features in the original space. The input to the neural network is the Base hypervectors ($\{B_1, \cdots, B_k\}$), and the output is the encoded data $\vec{\mathcal{H}}$. By training the network, PRID finds the best weights that generate the encoded hypervector given the base hypervectors. The trained neural network weights will be our decoded features.

Figure 1 visually compares analytical and learning-based data decoding over a few samples. Our evaluation shows that our learning-based solution is significantly more accurate in decoding data back to the original space. For example, for MNIST data points with 20% Gaussian noise ($\mu = 0, \sigma^2 = 1$), the analytical model provides $14.3dB$ Peak signal-to-noise ratio (PSNR) while learning-based model achieves PSNR of $29.1dB$. Recent work in [25] used an analytical model to show the differential privacy of the HDC model. The paper claimed that the decoded image from high-dimensional space provides very poor PSNR when adding a certain amount of noise to data. In contrast, our learning-based model can accurately recover such noisy encoded data. In fact, to ensure differential privacy, we require much larger noise that can come with significant overhead on the accuracy. Differential privacy is our of this paper scope, so we do not further explore it.

B. Train Data Reconstruction

The proposed data recovery approaches can be used to decode class hypervectors back to the original space. The decoding reveals the general shape of train data used for model training (e.g., shape of zero digits on MNIST). However, it is not clear if it reveals the information of train data used for model generation. In this section, we propose PRID that not only checks the membership of query data in a train set, but also reconstructs a good estimation of train data that has been used for model generation. Let us assume $\{\vec{T}_1, \vec{T}_2, \cdots, \vec{T}_p\}$ are p training data point in original space. HDC maps train data into high-dimensional space $\{\vec{\mathcal{H}}_{T1}, \vec{\mathcal{H}}_{T2}, \cdots, \vec{\mathcal{H}}_{Tp}\}$ and combines them to generate a model. For a classification problem with k classes, HDC generates k class hypervectors $\{\vec{\mathcal{C}}_1, \vec{\mathcal{C}}_2, \cdots, \vec{\mathcal{C}}_k\}$. We check the availability of a data point in a training set. For a given inference data, $\vec{F} = \{f_1, f_2, \cdots, f_n\}$, PRID first encodes the query data into high-dimensional space ($\vec{\mathcal{H}}$) using the same encoder used for training. Thereafter, we compute the cosine similarity (δ) of $\vec{\mathcal{H}}$ and all class hypervectors to find a class with the highest similarity: $\delta_{max} = \delta_l = \delta(\vec{\mathcal{H}}, \vec{\mathcal{C}}_l)$. For example, $\delta_l = 80\%$ indicates the there might be data points in training data that have high overlap with the query data. The next and more important question is to see if it is possible to recover the similar data points that have been used to train such a model. Here, we introduce two techniques to reconstruct data with high similarity to a query that has been used for training.

1) Feature Replacement: Figure 2 shows an overview of PRID data reconstruction. In the first step, PRID checks the existence of each element, e.g., feature, of the query data in the model (Figure 2a). To this end, PRID removes one of the features from the query data and encodes the new masked data into high-dimensional space (Figure 2b). For example, when we remove i^{th} feature from the query data, $\{f_1, \cdots, f_{i-1}, 0, f_{i+1}, \cdots, f_n\}$, the encoded hypervector represents as $\vec{\mathcal{H}}'^i$. We check the similarity of new encoded data with $\vec{\mathcal{C}}_l$, which is a class that the query belong to: $\delta_l^i = \delta(\vec{\mathcal{C}}_l, \vec{\mathcal{H}}'^i)$. If $\delta_l^i \geq \delta_{max} - \sigma$, it indicates that the i^{th} feature may exists in some training samples that used for $\vec{\mathcal{C}}_l$ model learning (Figure 2c). However, $\delta_l^i \leq \delta_{max} - \sigma$ indicates that the masked feature does not likely belong to any training sample. Note that σ^2 is variance over $\{\delta^1, \delta^2, \cdots, \delta^n\}$, and is considered as a similarity margin. PRID performs the same experiments over all features by removing them one-by-one from the original space and checking their existence on the model. This process selects all features of query data that are likely belong to training samples. For all other features, PRID replaces them with the corresponding features of the decoded class hypervector. As shown by Figure 2b, the class features can be computed by decoding the selected class back to the original space ($\vec{F}^C = \{f_n^C \cdots, f_1^C\}$). This results in generating a reconstructed data, $\vec{R}$ vector, which has combination of the query and class features.

$$\vec{R} = \langle r_n, \cdots, r_1 \rangle \quad \text{where} \quad r_i = \begin{cases} f_i & \text{if } \delta^j > \delta - \sigma^2 \\ f_i^c & \text{otherwise.} \end{cases} \tag{1}$$

where σ^2 is variance over $\{\delta^1, \delta^2, \cdots, \delta^n\}$. PRID performs an iterative method to improve the quality of data reconstruction. PRID encodes the reconstructed data, $\vec{R}$, to high-dimensional space, $H^{(1)}$, and compute its similarity to a corresponding class hypervector that a query belongs to, $\delta_{max}^{(1)} = \delta_l^{(1)} = \delta(\vec{\mathcal{C}}_l, \vec{\mathcal{H}}^{(1)})$. Then, PRID checks the existence of each feature of $\vec{R}$ in the model using the Equation 1. For features that do not exist in the model (result in a reduction in the similarity value), PRID replaces them with the inference data or class features. The replacement depends if the selected feature already corresponds to a query data or the class. If the selected features belong to the class, it will be replaced with the query features; otherwise, it will be replaced by the class features. After performing a few iterations of feature adjustment, PRID finds a reconstructed data with very high similarity to the training sample used for learning.

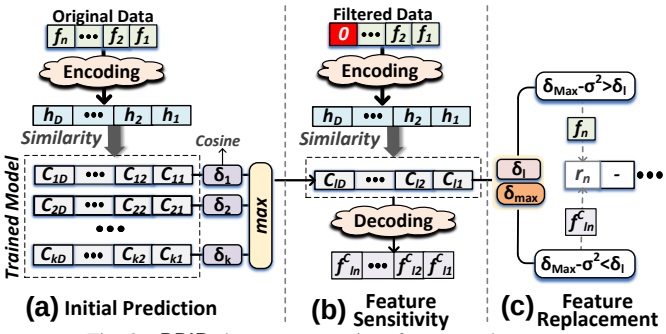


Fig. 2. PRID data reconstruction: feature replacement.

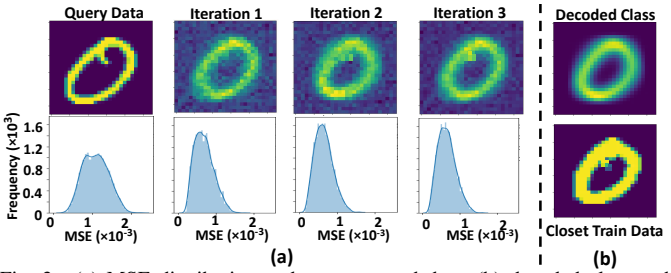


Fig. 3. (a) MSE distribution and reconstructed data, (b) decoded class and the closest train data to reconstructed data.

2) Dimension Replacement: Instead of replacing features in original space, we propose an approach that reconstructs train data by replacing high-dimensional features. For a query data F, and encoded hypervector $\vec{\mathcal{H}} = \{h_1, \cdots, h_D\}$, we first find a class with the highest similarity: $\delta_l = \delta(\vec{\mathcal{C}}_l, \vec{\mathcal{H}}_i)$. Then, we remove a single dimension from the encoded query data and check its similarity with the corresponding class hypervector ($\vec{\mathcal{C}}_l$). For example, when removing i^{th} dimension, we can compute the similarity of a new encoded data with selected class hypervector: $\delta_{max} = \delta_l^i = \delta_l^i(\vec{\mathcal{C}}_l, \vec{\mathcal{H}}_i)$. The term '$\delta_l^i \geq \delta_{max} - \sigma$' suggests that the i^{th} feature exists in the model, while $\delta_l^i \leq \delta_{max} - \sigma$ shows that the i^{th} dimension does not likely belong to any train samples. For every dimension that does not increase the similarity, we replace those dimensions with the class dimensions. This results in generating a new high-dimensional vector that has a combination of inference and class dimensions. Finally, we decode the combined high-dimensional vector back to the original space. Similar to Section III-B1, PRID improves the quality of the reconstructed data by performing an iterative approach. In each iteration, PRID encodes the new reconstructed data into high-dimensional space and finds dimensions that do not belong to the corresponding class hypervector. Finally, PRID reconstruct new data by replacing those dimensions. This process continues until finding a hypervector that the majority of dimensions have a high impact on the similarity.

Figure 3 visually shows a query sample, decoded class hypervector, and the reconstructed data using dimension replacement. To see the effectiveness of our approach, we check the similarity of inference and reconstructed data with all training data in original space, $\{T_1, T_2, \cdots, T_p\}$. Figure 3a shows the distribution of Mean Square Error (MSE) of train data with a query and reconstructed data. The results are reported for different iterations. Our evaluation shows that reconstructed data provides lower MSE as compared to query, suggesting that PRID is capable of extracting information from the trained model. Figure 3b also shows the best train data that matches with the reconstructed data. Our results indicates a high similarity of

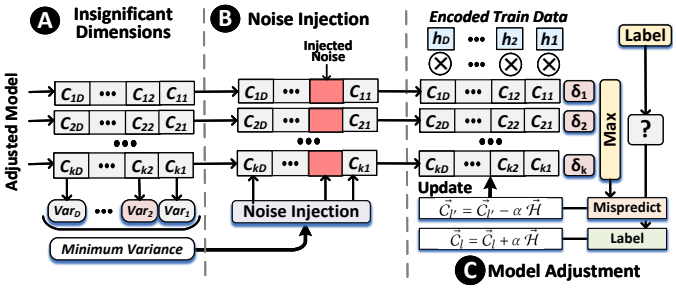


Fig. 4. PRID model privacy: intelligent noise injection.

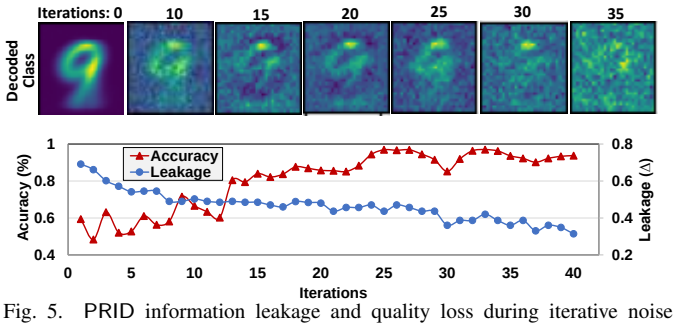


Fig. 5. PRID information leakage and quality loss during iterative noise injection procedure.

this data to our reconstructed data.

IV. PRID Privacy Preserving

We propose two techniques to increase PRID privacy against model inversion attack. Both methods aim to remove the shared information from the model to ensure privacy while having no or minimal affect on the accuracy.

A. Iterative & Intelligent Noise Injection

To improve the privacy, we propose the idea of intelligent noise injection that securtizes the HDC model by randomizing the insignificant model information. Figure 4 shows the overview of our framework, working in the following steps:

(i) Initial model training: it creates an initial model by accumulating all encoded data points corresponding to each class, $\{C_1, C_2, \cdots, C_k\}$.

(ii) Model decoding: PRID decodes the generated model back to original space using the learning-based method introduced in Section III-A, $\{\vec{F}_1^C, \vec{F}_2^C, \cdots, \vec{F}_k^C\}$, where $\vec{F}_i^C \in \mathbb{R}^n$.

(iii) Noise Injection: PRID identifies the insignificant class elements in hyperdimensional space that store common information. These dimensions have minor impact on the classification accuracy. As Figure 4a shows, these features are obtained by computing the variance over different class features in the adjusted model. PRID replaces those features with a random noise that has the same distribution as other features.

(iv) Model Adjustment: the noise injection can possibly result in a quality loss. PRID compensates this loss by retraining the noisy model over the train data (Figure 4c). PRID checks the similarity of an encoded train data with the noisy HDC model. If the model mispredicts data (which corresponding to label l) as label l', the model updates as:

$$\vec{C}_l = \vec{C}_l + \alpha\, \vec{\mathcal{H}} \qquad \vec{C}_{l'} = \vec{C}_{l'} - \alpha\, \vec{\mathcal{H}} \tag{2}$$

where α is a learning rate. This process continues over all train data to create a bigger gap between the classes.

(v) Iterative Learning: PRID continues the noise injection (Step III) and model adjustment (step II) until the HDC accuracy stabilizes during few consecutive iterations. Figure 5 shows the impact of iterative noise injection on the privacy and accuracy of the HDC model. In the first iteration, PRID has a deficient level of privacy as the HDC model reveals the class hypervectors. Noise injection improves model privacy by randomizing the information stored in the class hypervectors (40% in our example). The class hypervectors become more blurry after each retraining iteration, resulting in a higher privacy (Figure 5a). The retraining also compensates for the quality loss caused by noise injection (Figure 5b).

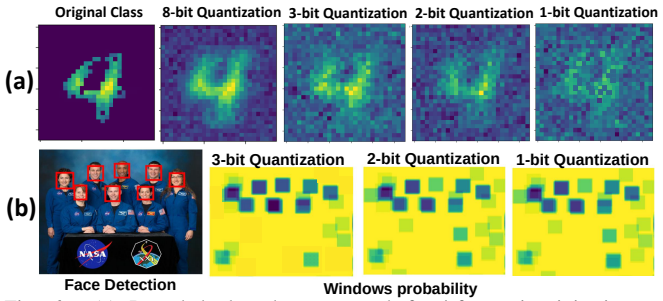


Fig. 6. (a) Decoded class hypervector before/after noise injection and quantization, (b) face detection with quantization.

B. Iterative Model Quantization

In HDC, the class hypervectors are represented using 32-bit values. PRID introduces model quantization as a practical approach to reduce the precision of each class dimension to n-bits ($n < 32$). Quantization reduces the chance of decoding the high-dimensional data back to original space, enhancing the model privacy. Our framework integrates the quantization as a part of an iterative training to: (i) teach the HDC model to work with the quantization constraints, and (ii) further separate the class hypervectors and improve the model privacy. Our model quantization implements in the following steps:

(i) Initial training: the initial training is performed using the same method as Step I in Section IV-A. **(ii) Model quantization:** PRID quantizes the trained model and stores two copies of the HDC models: full precision and quantized models (n-bit precision).

(iii) Model adjustment: PRID checks the similarity of the encoded data with the quantized model. For each misprediction, PRID uses Equation 2 to only update the full precision model. The update on the quantized model can result in a divergence since the quantized model does not have enough precision for an update.

(iii) Model Update: After updating the model over all training data (or batch of data), PRID quantizes the full-precision model and replaces it with the quantized model. Figure 6a shows the decoded class hypervectors after different model quantization. The result shows that quantization improves model privacy by lowering the amount of information stored in the model. However, the naive model quantization can result in significant quality loss.

(iv) Iterative learning: To compensate for the possible quality loss, PRID iteratively continues the above steps until the accuracy stabilizes over a few consecutive iterations. Figure 6b shows the face detection accuracy (the probability windows) after model quantization. Our evaluation shows that the quality of face detection decreases by the quantization. For example, 1-bit (2-bit) model quantization results in 4.8% (3.3%) quality loss as compared to full precision model. The level of quantization affects both accuracy and privacy. Using extreme quantization ($n = 1$), PRID has the highest level of privacy while it also has the highest quality loss during the classification (Section V-D).

TABLE I
DATASETS (n: FEATURE SIZE, k: NUMBER OF CLASSES).

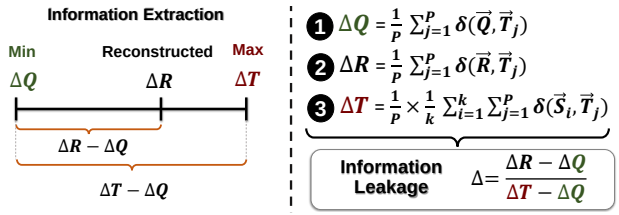
	n	K	Train Size	Test Size	Description/State-of-the-art Model
SPEECH	617	26	6,238	1,559	Voice Recognition/DNN [31]
MNIST	784	10	50,000	10,000	Handwritten digits/DNN [32]
FACE	608	2	522,441	2,494	Face recognition/Adaboost [33]
ACTIVITY	75	5	611,142	101,582	Activity recognition(IMU)/DNN [30]
EXTRA	225	4	146,869	16,343	Phone position recognition/AdaBoost [34]
UCIHAR	561	12	6,213	1,554	Activity recognition(Mobile)/DNN [35]

V. EVALUATION

We verified PRID functionality using Python implementation. We also provide open-source Python implementation. The tested benchmarks range from relatively small datasets collected in a small IoT network, e.g., ACTIVITY [30], to a large dataset that includes hundreds of thousands of images of facial and non-facial data. Table I also lists the model for each dataset. Comparing the accuracy results, we observe that PRID provides comparable accuracy to state-of-the-art algorithms (only 0.2% lower average accuracy).

We define information leakage by computing the similarity of reconstructed data with the training data set. For a query $\vec{\mathcal{Q}}$ and reconstructed data $\vec{\mathcal{R}}$, it defines as:

- *Minimum extracted information* from a model (ΔQ), which can be computed as the average similarity between an unbiased constant vector $G = (1, 1, \ldots, 1)$ with the entire training set.
- *Maximum extracted information* from the model (ΔT). This can be computed by: (i) finding top-k values in train data (original space) with the highest similarity to the query (S), (ii) computing the average similarity of those values from the entire train set.
- *Reconstructed information extraction* from a model (ΔR), which can be computed by the average similarity of reconstructed data from the entire train set.



As above equation shows, the reconstructed data extracts a value between the minimum (ΔQ) and the maximum possible ΔT information extraction. A large Δ indicates that the reconstructed data has a higher similarity to the train set; thus, the model has high information leakage. In contrast, smaller Δ shows the closeness of the query and reconstructed data.

A. Privacy Attack in HDC

Figure 7 compares the effectiveness of two proposed data reconstruction methods: feature and dimension replacement. The results are reported using both analytical and learning-based data decoding, introduced in Section III-A. Our evaluation shows that our learning-based solution always outperforms the analytical approach in terms of the quality of data decoding. Our results also indicate that feature-based reconstruction provides higher information leakage (Δ) compared to dimension-based. In contrast, dimension-based reconstruction provides higher PSNR. Depending on the definition of privacy, both these approaches can effectively extract information about the train set just by accessing the model. To extract maximum information from the train set, we combined both feature-based and dimension-based techniques. In every iteration, PRID first reconstruct an input using feature-based while in the next iteration PRID uses dimension-based reconstruction. As Figure 7 shows, this approach

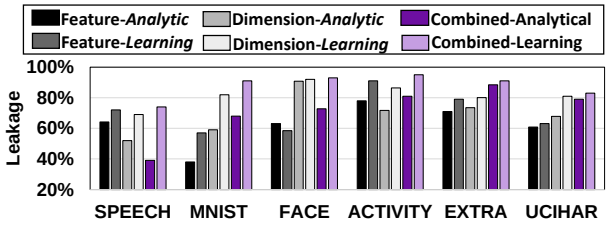


Fig. 7. Privacy using different data reconstructions.

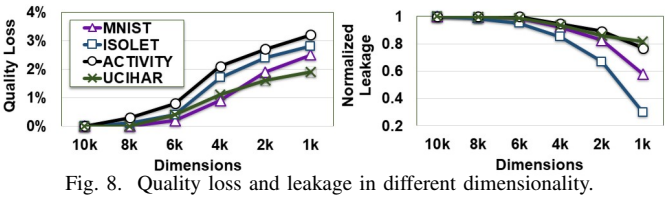


Fig. 8. Quality loss and leakage in different dimensionality.

reveals maximum information from train data, results in maximum privacy concerns. For the rest of the evaluation, we consider this combinational data reconstruction to attack the model privacy.

B. Dimensionality & Privacy-Accuracy

Figure 8 shows PRID data reconstruction and the quality loss using hypervectors with different dimensionality. The quality loss and information leakage are reported compared to PRID using $D = 10k$ dimensions. A hypervector with higher dimensions can store more useful information; thus, it provides a higher chance for data reconstruction. Although reducing dimensionality is a practical approach to degrade the data reconstruction rate, aggressive dimension reduction can result in possible quality loss. For example, our evaluation indicates that PRID using 2k (1k) dimensionality reduces the information leakage to 81% (62%) while resulting in less than 2.1% and 2.4% quality loss on classification accuracy.

C. PRID Noise Injection

Figure 9 shows the impact of noise injection on PRID accuracy and privacy. The results are reported when injecting noise into a different portion of features on the decoded model. For example, 20% noise indicates that we randomize 20% of the decoded class features with the lowest variance. In PRID, there is a trade-off between the model accuracy and information leakage. Adding noise degrades the classification accuracy while it improves the model privacy by vanishing the actual information from the model. Our evaluation shows that HDC provides high robustness to the noise injection, because (i) PRID only adds noise to insignificant features that have minor impact on the classification, (ii) HDC has inherent robustness to noise and failure in dimensions, and (iii) thanks to our retraining method that compensate for the quality loss. For example, randomizing 20% and 60% features only results in 3.5% and 9.6% average quality loss. Note that without retraining, these injected noise values could result in 12.7% and 48.1% quality loss, which is not acceptable in practice. Increasing the level of noise features improves PRID privacy by making decoding nearly impossible. Our results indicate that using 20% and 60% noisy features improves PRID privacy by 20.9% and 43.3%, respectively.

D. PRID Model Qunatization

The privacy of HDC against model inversion attack directly relates to the quantization level. The quantization reduces the amount of useful information stored in each class hypervector. Figure 10 shows the information leakage (Δ) of the HDC model when the level of quantization varies from 1-bit to 32-bits. Our evaluation shows that

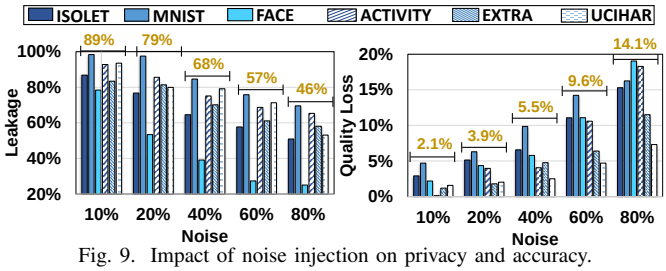


Fig. 9. Impact of noise injection on privacy and accuracy.

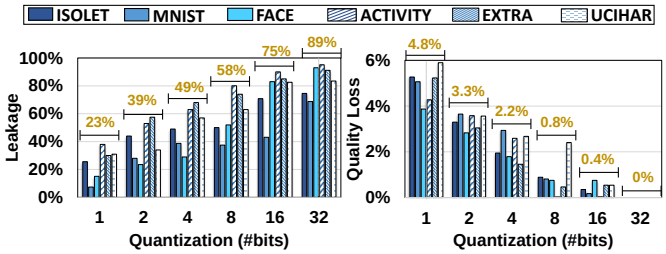


Fig. 10. Impact of quantization on privacy and accuracy.

model quantization to 1-bit and 4-bits reduces the information leakage to 86.9% and 51.2%, respectively. This indicates that the quantized model has poor decoding, which does not help reconstruct training data. As explained in Section IV-B, the naive quantization can result in significant quality loss in HDC. However, our proposed iterative framework ensures that the model learns to work with quantization constraints with minimal loss. Our evaluation shows that PRID provides maximum accuracy using 4-bit quantization. Reducing the quantization level to 1-bit (4-bit) results in an average of 4.8% and 2.2% quality loss.

E. Hybrid: Quantization & Noise Injection

To enhance the level of privacy, we combine the noise injection and quantization method. During iterative training, PRID stores both full-precision and quantized model. In each iteration of model adjustment, PRID injects noise to the full-precision model. During iterative training, PRID checks the similarity with the quantized model while updates the full precision model for each misprediction. After going through all train data (or batch of data), PRID updates both models by quantizing the noisy model. Our framework iteratively retrains the model to teach the HDC model to work with noise injection and quantization constraints. Table II shows the information leakage that each approach provides in different quality loss. Our result indicates that, at the same level of accuracy, the combined approach results in minimum information leakage.

VI. Conclusion

In this paper, we first show the possibility of a model inversion attack in hyperdimensional computing. We exploit invertiblity of the HDC encoding module to expose the HDC model and extract useful information from it. We propose two iterative techniques which scrutinize HDC model from a privacy perspective: (i) intelligent noise injection that identifies and randomizes insignificant features of the model in the original space, and (ii) model quantization that removes model's recoverable information while teaches the model iteratively to compensate the possible quality loss. We evaluate PRID effectiveness over a wide range of classification problems. Our framework enables HDC to provide high accuracy while minimizing the information leakage.

Acknowledgment

This work was partially supported by Department of the Navy, Office of Naval Research, grant #N000142112225 (ONR Program Manager: Sarwat Chappell) and Semiconductor Research Corporation (SRC) Task No. 2988.001.

TABLE II
Information leakage in different quality loss.

Quality loss	@0.5%	@1%	@2%	@3%	@5%
Noise Injection	7%	11%	15%	22%	32%
Quantization	15%	32%	48%	59%	87%
Combined	19%	35%	53%	66%	92%

References

[1] M. Medwed, "Iot security challenges and ways forward," in *TrustED*, 2016.
[2] X. Gu and A. Easwaran, "Towards safe machine learning for cps: infer uncertainty from training data," in *ICCPS*, pp. 249–258, 2019.
[3] S. Ghandali *et al.*, "Side-channel hardware trojan for provably-secure sca-protected implementations," *TVLSI*, vol. 28, no. 6, pp. 1435–1448, 2020.
[4] M. Conti *et al.*, "Do we need a holistic approach for the design of secure iot systems?," in *Proceedings of the Computing Frontiers Conference*, pp. 425–430, 2017.
[5] J. Fan and I. Verbauwhede, "An updated survey on secure ecc implementations: Attacks, countermeasures and cost,"
[6] M. Striegel *et al.*, "Secure and user-friendly over-the-air firmware distribution in a portable faraday cage," in *WiSec*, pp. 173–183, 2020.
[7] I. Vistbakka and E. Troubitsyna, "Pattern-based formal approach to analyse security and safety of control systems," in *IMBSA*, pp. 363–378, Springer, 2019.
[8] D. De Niz *et al.*, "Mixed-trust computing for real-time systems," in *RTCSA*, pp. 1–11, IEEE, 2019.
[9] R. I. Davis *et al.*, "A survey of probabilistic schedulability analysis techniques for real-time systems," *LITES*, pp. 1–53, 2019.
[10] A. S. Krishnan, C. Suslowicz, and P. Schaumont, "Secure and stateful power transitions in embedded systems," *HaSS*, pp. 1–14, 2020.
[11] M. Denil *et al.*, "Predicting parameters in deep learning," in *NISP*, 2013.
[12] A. Zaslavsky *et al.*, "Sensing as a service and big data," *arXiv preprint arXiv:1301.0159*, 2013.
[13] Y. Sun *et al.*, "Internet of things and big data analytics for smart and connected communities," *IEEE Access*, vol. 4, pp. 766–773, 2016.
[14] Y. Xiang and H. Kim, "Pipelined data-parallel cpu/gpu scheduling for multi-dnn real-time inference," in *RTSS*, pp. 392–405, IEEE, 2019.
[15] B. Babadi *et al.*, "Sparseness and expansion in sensory representations," *Neuron*, vol. 83, no. 5, pp. 1213–1226, 2014.
[16] A. Hernandez-Cano *et al.*, "A framework for efficient and binary clustering in high-dimensional space," in *DATE*, IEEE, 2021.
[17] M. Imani, S. Salamat, B. Khaleghi, M. Samragh, F. Koushanfar, and T. Rosing, "Sparsehd: Algorithm-hardware co-optimization for efficient high-dimensional computing," in *2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pp. 190–198, IEEE, 2019.
[18] M. Imani *et al.*, "Quanthd: A quantization framework for hyperdimensional computing," *TCAD*, 2019.
[19] A. Hernandez-Cano *et al.*, "Onlinehd: Robust, efficient, and single-pass online learning using hyperdimensional system," in *DATE*, IEEE, 2021.
[20] M. Imani *et al.*, "Dual: Acceleration of clustering algorithms using digital-based processing in-memory," in *MICRO*, pp. 356–371, IEEE, 2020.
[21] Z. Zou *et al.*, "Manihd: Efficient hyper-dimensional learning using manifold trainable encoder," in *DATE*, IEEE, 2021.
[22] M. Imani, A. Rahimi, D. Kong, T. Rosing, and J. M. Rabaey, "Exploring hyperdimensional associative memory," in *High Performance Computer Architecture (HPCA), 2017 IEEE International Symposium on*, pp. 445–456, IEEE, 2017.
[23] F. Yang and S. Ren, "Adversarial attacks on brain-inspired hyperdimensional computing-based classifiers," *arXiv preprint arXiv:2006.05594*, 2020.
[24] M. Imani *et al.*, "A framework for collaborative learning in secure high-dimensional space," in *CLOUD*, pp. 435–446, IEEE, 2019.
[25] B. Khaleghi *et al.*, "Prive-hd: Privacy-preserved hyperdimensional computing," *arXiv preprint arXiv:2005.06716*, 2020.
[26] M. Imani *et al.*, "Hierarchical hyperdimensional computing for energy efficient classification," in *DAC*, p. 108, ACM, 2018.
[27] M. Imani *et al.*, "Revisiting hyperdimensional learning for fpga and low-power architectures," in *HPCA*, IEEE, 2021.
[28] M. Fredrikson *et al.*, "Model inversion attacks that exploit confidence information and basic countermeasures," in *CCS*, pp. 1322–1333, 2015.
[29] Y. Wang *et al.*, "Regression model fitting under differential privacy and model inversion attack," in *IJCAI*, 2015.
[30] A. Reiss *et al.*, "Introducing a new benchmarked dataset for activity monitoring," in *ISWC*, pp. 108–109, IEEE, 2012.
[31] "Uci machine learning repository." http://archive.ics.uci.edu/ml/datasets/ISOLET.
[32] Y. LeCun, C. Cortes, and C. J. Burges, "Mnist handwritten digit database," *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2010.
[33] Y. Kim *et al.*, "Orchard: Visual object recognition accelerator based on approximate in-memory processing," in *ICCAD*, pp. 25–32, IEEE, 2017.
[34] Y. Vaizman *et al.*, "Recognizing detailed human context in the wild from smartphones and smartwatches," *Pervasive Computing*, vol. 16, no. 4, pp. 62–74, 2017.
[35] D. Anguita *et al.*, "Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine," in *AAL*, pp. 216–223, Springer, 2012.