MEMBERSHIP INFERENCE ATTACKS ON HYPERDIMENSIONAL COMPUTING: A COMPREHENSIVE ARCHITECTURAL ANALYSIS

RUPESH RAJ KARN, JOHANN KNECHTEL, OZGUR SINANOGLU
Center for Cyber Security, New York University, Abu Dhabi, UAE
E-MAIL: {rupesh.k, johann, ozgursin}@nyu.edu

Abstract:

Membership Inference Attacks (MIAs) pose a significant threat to the privacy of machine learning (ML) models by allowing adversaries to determine whether a specific data sample was part of a model's training dataset. While extensive literature has explored MIAs in traditional ML models such as neural networks and decision trees, little attention has been given to their applicability in the emerging paradigm of Hyperdimensional Computing (HDC). This work pioneers the systematic exploration of MIAs in HDC models, which are increasingly being deployed in domains ranging from image classification to hardware security. We evaluate the susceptibility of HDC to MIAs using two diverse datasets: the MNIST image dataset and digital circuit netlists from ISCAS85 and EPFL combinational benchmarks. We leverage multiple similarity-based metrics—Cosine, Euclidean, and Hamming distance—to distinguish training members from non-members. Our results demonstrate that the direct aggregation and transparent structure of HDC models make them more vulnerable to MIA compared to traditional ML models, which often abstract and obscure individual training contributions.

Keywords:

Hyperdimensional Computing, Membership Inference Attacks, Privacy, High-Dimensional Vectors, Model Vulnerability

1 Introduction

Machine learning (ML) models are increasingly being scrutinized for their ability to inadvertently leak information about their training data [?]. Among these privacy risks, Membership Inference Attacks (MIAs) [?, ?] have emerged as a potent vector, allowing adversaries to infer whether a particular data point was used during a model's training phase. This concern is particularly amplified in sensitive domains such as healthcare, security, and finance, where training membership itself may carry confidential information.

While MIAs have been well-studied in traditional ML paradigms such as neural networks (NNs) and decision trees (DTs) [?], these attacks rely on the presence of overfitting, confidence score anomalies, or complex behavior in model outputs [?]. Such methods assume a parametric model with hidden, non-linear decision boundaries, and often require access to probability scores or softmax vectors to be effective. However, these assumptions do not hold in the emerging class of models based on Hyperdimensional Computing (HDC) [?, ?].

HDC models operate on high-dimensional binary or bipolar hypervectors and rely on simple arithmetic and logical operations for both training and inference [?, ?]. Class representations in HDC are formed through additive accumulation of encoded data samples, making them inherently transparent and memory-based [?]. As HDC finds increasing use in diverse domains—from image classification tasks like MNIST to security-critical applications involving ISCAS85 and EPFL digital circuit netlists—it becomes imperative to investigate whether MIAs can compromise their privacy guarantees.

Here, we present the first comprehensive exploration of MIAs on HDC models. We introduce a set of tailored methodologies for executing MIAs against HDC systems, taking into account their unique vector-based architecture and similarity-driven inference mechanism. Unlike traditional models, HDC exposes explicit data influence in its memory model [?], raising critical questions about the privacy resilience of such systems.

The main contributions of this work are as follows:

1. We present the first dedicated study of MIAs on HDC models, highlighting their vulnerability due to explicit training sample representation.
2. We conduct extensive experiments on two domains: the MNIST image classification dataset and the ISCAS85 and EPFL combinational circuit netlist datasets, demonstrating the generalization of our find-

ings.

3. We compare the effectiveness of MIAs against HDC models with traditional models like NNs and DTs, showcasing the heightened risk in HDC due to its deterministic and memory-based structure.

Source code to reproduce results is available at https://github.com/rkarn/Membership_Inference_Attack .

2 Preliminaries

This section covers core concepts, beginning with motivation, followed by the HDC model's mathematical foundations and MIA fundamentals, establishing the basis for our methodology.

2.1 Motivation

Machine learning models are increasingly deployed in privacy-sensitive domains including healthcare diagnostics, financial fraud detection, and personalized services, exposing critical vulnerabilities to MIAs [?]. Unlike traditional cyberattacks, MIAs determine whether specific data was used to train a model, with serious consequences. In healthcare, inferring a patient's data was used in training could reveal their medical conditions. For financial datasets, MIAs may expose sensitive transaction patterns. These attacks can violate data protection laws like GDPR [?] and HIPAA [?], and often enable more sophisticated attacks like data poisoning and model inversion.

While existing research has extensively studied MIAs on NNs, support vector machine (SVMs), and random forests (RFs), the vulnerability of HDC models remains unexplored. This gap is particularly concerning as HDC's training explicitly accumulates samples in class hypervectors, potentially making membership information more accessible than in NN representations. Additionally, the paradigm's growing adoption in edge devices and IoT applications—where models are often deployed with minimal protection—raises urgent privacy concerns [?, ?]. Further, HDC's binary/bipolar vector space may enable novel attack vectors differing from conventional ML models.

Our work addresses this oversight through the first systematic investigation of MIAs against HDC systems, providing insights for both attackers and defenders. Also see our release for detailed comparison with prior arts.

2.2 HDC Model

Hyperdimensional computing is a biologically inspired computational paradigm in which data is represented using high-dimensional vectors, known as hypervectors. These are typically bipolar vectors in $\{-1,+1\}^D$, where $D \gg 1$ (e.g. $D = 10{,}000$).

Memory and Encoding: Each input feature f_i and its quantized value v_i are mapped to hypervectors via an Item Memory (IM) and Value Memory (VM), respectively. Given a sample $\mathbf{x} = [x_1, x_2, \ldots, x_n]$, the encoded hypervector $\mathbf{h}_x \in \{-1,+1\}^D$ is constructed using binding and bundling operations:

$$\mathbf{h}_x = \sum_{i=1}^{n} \rho_i \left(\mathbf{h}_{f_i}\right) \circ \mathbf{h}_{v_i}, \tag{1}$$

where $\mathbf{h}_{f_i} = \mathrm{IM}(f_i)$ is the hypervector for the i-th feature, $\mathbf{h}_{v_i} = \mathrm{VM}(v_i)$ is the hypervector for the quantized feature value, $\circ$ denotes element-wise multiplication (binding), and $\rho_i(\cdot)$ is a fixed permutation that encodes feature position.

Training via Hypervector Aggregation: The class hypervector $\mathbf{H}_c$ for class c is formed by aggregating encoded hypervectors from all samples $\mathbf{x}_j \in \mathcal{D}_c$ belonging to class c:

$$\mathbf{H}_c = \sum_{\mathbf{x}_j \in \mathcal{D}_c} \mathrm{Encode}(\mathbf{x}_j). \tag{2}$$

Inference via Similarity: For inference, an input sample $\mathbf{x}_q$ is encoded into $\mathbf{h}_q = \mathrm{Encode}(\mathbf{x}_q)$, and classification is based on similarity between $\mathbf{h}_q$ and each class hypervector:

$$\hat{c} = \arg\max_c \left(\mathrm{sim}(\mathbf{h}_q, \mathbf{H}_c)\right), \tag{3}$$

where $\mathrm{sim}(\cdot,\cdot)$ denotes cosine similarity:

$$\mathrm{sim}(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|} \tag{4}$$

2.3 MIA

An MIA aims to infer whether a specific data point $\mathbf{x}_q$ was included in the training dataset of a target model. MIAs exploit patterns, overfitting behavior, or output distributions to infer membership, thus compromising data privacy. Formally, an adversary seeks to determine the binary membership indicator:

$$M(\mathbf{x}_q) = \begin{cases} 1, & \text{if } \mathbf{x}_q \in \mathcal{D}_{\text{train}}, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Given a query sample $\mathbf{x}_q$, the attacker encodes it to obtain $\mathbf{h}_q$, and computes its similarity to the class hypervector $\mathbf{H}_c$. If the similarity exceeds a chosen threshold τ, the adversary infers that $\mathbf{x}_q$ was used during training:

$$\mathrm{sim}(\mathbf{h}_q, \mathbf{H}_c) > \tau \Rightarrow \mathbf{x}_q \in \mathcal{D}_c. \tag{6}$$

This inference is justified by the additive nature of class hypervector $\mathbf{H}_c$ (Equation 2): training samples contribute directly to the hypervector, making their encoded representations more similar than those of non-members.

3 MIA to HDC

The HDC models warrant a fresh look with respect to MIA. The architectural properties of HDC models introduce unique attack surfaces for MIAs. These vulnerabilities stem from three fundamental characteristics:

3.1 Additive Training Mechanism

Supervised HDC models construct class representations through direct aggregation of encoded samples (Equation 2). This operation preserves exact geometric relationships between samples and class centroids, maintains linear separability of contributions (unlike non-linear NN transformations), and permits sample influence to be provably bounded via vector algebra.

3.2 Transparent Data Representation

Traditional ML models abstract training data through non-linear activation functions in NNs, ensemble methods in RFs, and kernel transformations in SVMs. In contrast, HDC's memory-based paradigm retains direct traceability of training samples. Each $\mathbf{x}_i$'s hypervector contributes equally to $\mathbf{H}_c$ modulo the encoding function's properties.

3.3 Distance-Based Inference

HDC's inference relies on similarity metrics (Equation 4), which enables adversaries to compute exact membership scores using the same metrics as legitimate inference, to establish threshold-based detectors (Equation 6) without requiring probability calibration, and to bypass the need for auxiliary shadow models (unlike with MIAs on NNs [?]).

The comparison is summarized in Table 1. This qualitative analysis suggests that HDC models are intrinsically more vulnerable to MIA due to their deterministic properties that preserve distance. The absence of non-linear transformations or stochastic regularization mechanisms makes membership signals fundamentally harder to obscure compared to traditional ML approaches. In later sections of this work, we present experimental evidence to illustrate this aspect.

4 Different Architectures of HDC

Although standard HDC models (explained in Section 2.2) offer a robust and interpretable mechanism for learning, their performance can be limited in high-variability or high-dimensional datasets. To address this, several modifications have been applied to the HDC architecture and configurations that not only improve accuracy but also impact the susceptibility and evaluation strategy for MIAs. Some of those architectures and configurations are as follows.

4.1 Iterative Retraining and Regenerative Training

In standard HDC, the class hypervector $\mathbf{H}_c$ is formed as an unweighted sum of encoded training samples as in Equation 2. However, this initial model may misclassify samples near decision boundaries. To correct this, an iterative retraining mechanism [?] is employed where misclassified samples are identified post-inference and reintroduced with higher weights:

$$\mathbf{H}_c^{(t+1)} = \mathbf{H}_c^{(t)} + \lambda \sum_{\mathbf{x}_k \in \mathcal{M}_c^{(t)}} \text{Encode}(\mathbf{x}_k), \tag{7}$$

where $\mathcal{M}_c^{(t)} \subset \mathcal{D}$ is the set of misclassified samples at iteration t predicted as class c, and $\lambda > 1$ is a learning rate or weight amplification factor.

This regenerative step enhances the representational capacity of class vectors for difficult regions in the input space, thereby improving classification accuracy [?].

4.2 Adaptive Encoding

Traditional HDC encodes features using randomly generated hypervectors, which may not fully exploit the underlying data structure. We use an adaptive encoding [?] mechanism that leverages Principal Component Analysis (PCA) or other linear transformations to emphasize the most informative features:

$$\mathbf{z}_j = \mathbf{W}_{\text{PCA}}^\top \mathbf{x}_j, \tag{8}$$

where $\mathbf{x}_j \in \mathbb{R}^n$ is the original input vector, and $\mathbf{W}_{\text{PCA}} \in \mathbb{R}^{n \times k}$ is the matrix of top k eigenvectors from PCA. The transformed vector $\mathbf{z}_j$ is quantized and encoded via standard HDC.

This dimensionality reduction improves signal-to-noise ratio and increases class separability in the hypervector space [?].

4.3 Hybrid HDC Models

To enhance feature representation, a hybrid architecture is used that integrates a lightweight NN f_θ to extract non-linear features [?], which are then encoded using the HDC framework:

$$\mathbf{h}_j = \text{Encode}(f_\theta(\mathbf{x}_j)), \tag{9}$$

where $f_\theta : \mathbb{R}^n \to \mathbb{R}^k$ is a neural feature extractor parameterized by θ. This model benefits from the expressive

TABLE 1. Comparison of MIA risks in HDC vs. traditional ML models.

Feature	Traditional ML Models	HDC Models
Abstraction of training data	High (deep architectures abstract individual samples)	Low (individual samples directly contribute to class representation)
Sensitivity to each sample	Typically low unless overfitted	High due to explicit accumulation of encoded vectors
Training data persistence	Implicit (via weights or rules)	Explicit (summed in class hypervectors)
Predictive mechanism	Parametric (e.g., learned weights or splits)	Similarity-based (cosine or Hamming to class vectors)
Complexity of inference logic	High (nonlinear functions, layers)	Low (vector similarity)
Ease of reverse engineering	Low (complex internal structure)	Moderate to High (transparent vector encoding)

capacity of NNs for complex data and the symbolic interpretability and efficiency of HDC in the decision layer.

These enhancements notably improve classification performance while also impacting the vector space topology by either increasing the distinctiveness of members (exacerbating MIA) or adding robustness that generalizes better, thereby reducing overfitting and lowering MIA risk. Therefore, any MIA strategy must account for the transformation and retraining dynamics introduced by these mechanisms.

4.4 Higher-Order Representations

To capture complex feature interactions beyond linear aggregation, a higher-order representations [?] is applied with the HDC encoding. Instead of encoding features independently, subsets of features are grouped and bound using element-wise operations such as multiplication (binding) or XOR:

$$\mathbf{h}_{i,j} = \mathbf{h}_{f_i} \circ \mathbf{h}_{f_j}, \quad \mathbf{h}_{\text{group}} = \sum_{(i,j)\in\mathcal{P}} \mathbf{h}_{i,j}, \tag{10}$$

where $\mathcal{P} \subseteq \{(i,j) : i < j\}$ is the set of feature pairs considered. This group hypervector $\mathbf{h}_{\text{group}}$ is then included in the class hypervector accumulation:

$$\mathbf{H}_c \leftarrow \mathbf{H}_c + \mathbf{h}_{\text{group}}. \tag{11}$$

This formulation allows the model to implicitly learn feature correlations and dependencies, which is especially useful in datasets with structured or hierarchical relationships [?].

4.5 Dimensionality Optimization

The dimensionality D of the hypervectors plays a crucial role in determining the representational capacity of the HDC model. A larger D provides more orthogonal space for representing patterns but also incurs higher computational and memory costs. We explore dimensionality values from $D = 10{,}000$ up to $D = 50{,}000$, optimizing for validation accuracy:

$$D^* = \arg\max_D \ \text{Accuracy}(\text{HDC}_D(\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}})). \tag{12}$$

Empirically, increasing D can reduce the overlap between encoded hypervectors, thereby increasing classification precision and robustness against adversarial noise.

4.6 Error-Correcting Codes

To enhance the robustness of class representations against misclassification and noise, the error-correcting codes (ECC) is applied [?] to class labels. Each class $c \in \{1, \ldots, C\}$ is mapped to a binary codeword $\mathbf{y}_c \in \{0,1\}^K$ using a pre-defined ECC scheme such as BCH or Hamming codes. The HDC model is trained to predict the codeword instead of the raw label:

$$\hat{\mathbf{y}} = \text{Decode}(\mathbf{h}_x), \tag{13}$$

where $\text{Decode}(\cdot)$ determines the most similar codeword using Hamming or cosine similarity. The final predicted class is obtained via ECC decoding:

$$\hat{c} = \text{ECC}^{-1}\left(\arg\min_c \text{dist}(\hat{\mathbf{y}}, \mathbf{y}_c)\right). \tag{14}$$

This approach introduces redundancy in the label space, which enhances resistance to encoding and decoding noise.

4.7 MicroHD Optimization

MicroHD [?] is an optimization-driven framework tailored to systematically tune the HDC pipeline for optimal performance. It involves hyperparameter exploration and dynamic adjustment of encoding strategies. Given a search space Θ of hyperparameters such as dimensionality, quantization levels, and encoding rules, MicroHD performs:

$$\theta^* = \arg\max_{\theta\in\Theta} \ \text{Accuracy}(\text{HDC}_\theta(\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}})), \tag{15}$$

where θ includes parameters such as the dimensionality D, the feature grouping structure for higher-order encoding,

and the binding operation (e.g., XOR vs. element-wise product).

By automating the design space exploration, MicroHD identifies the optimal configuration that maximizes model performance while balancing efficiency and robustness.

5 MIA Evaluation

5.1 Evaluation Metric

For each sample $\mathbf{x}_i$ with label y_i, we compute its encoded hypervector $\mathbf{h}_i = \text{Encode}(\mathbf{x}_i)$, and evaluate the similarity score s_i using one of the following metrics:

$$\text{Cosine Similarity: } s_i^{(\cos)} = \frac{\mathbf{h}_i \cdot \mathbf{H}_{y_i}}{\|\mathbf{h}_i\|\|\mathbf{H}_{y_i}\|} \quad (16)$$

$$\text{Euclidean Distance (negated): } s_i^{(\text{euc})} = -\|\mathbf{h}_i - \mathbf{H}_{y_i}\|_2 \quad (17)$$

$$\text{Hamming Distance (negated): } s_i^{(\text{ham})} = -d_{\text{ham}}(\mathbf{h}_i, \mathbf{H}_{y_i}) \quad (18)$$

The negation of Euclidean and Hamming distances serves to maintain interpretational consistency with cosine similarity.

A sample is inferred to be a member of the training set if its score exceeds a threshold τ calibrated from the training data (Equation 6). The membership inference can be rated by:

– True Positive Rate (TPR): Fraction of training samples correctly inferred as members;

– True Negative Rate (TNR): Fraction of test samples correctly inferred as non-members.

A High TPR indicates the model retains detectable traces of training samples, while a low TNR suggests overexposure of non-members. Together, they reveal the model's membership privacy gap—the disparity between how it treats training versus unseen data. By analyzing these rates across different metrics and HDC configurations (Section 4), we assess the degree to which the model retains information about its training data, revealing potential privacy risks.

5.2 Threshold Selection for MIA

The threshold τ is calibrated using the average training score (μ_{train}) for each metric, computed as the arithmetic mean of all training samples' similarity/distance scores:

$$\tau_{\text{cosine}} = \mu_{\text{train}}^{\text{cosine}} + 0.02, \quad (19)$$

$$\tau_{\text{euclidean}} = \mu_{\text{train}}^{\text{euclidean}}, \quad (20)$$

$$\tau_{\text{hamming}} = \mu_{\text{train}}^{\text{hamming}}. \quad (21)$$

For cosine similarity, we add a small, empirically derived margin to account for its saturated distribution near 1. This thresholding strategy is designed to exploit the inherent separation between member and non-member distributions—where training samples typically exhibit higher cosine similarity scores or smaller distances for Euclidean/Hamming distances—and to maintain robustness over score variations across HDC architectures through metric-specific calibration.

5.3 Experimental Testbench

To comprehensively evaluate the susceptibility of HDC models to MIAs, we design a testbench that includes diverse datasets, custom preprocessing pipelines, and standardized ML baselines. The experiments are implemented in Python and executed in a high-performance computing environment to ensure reproducibility and scalability.

We utilize two significantly different datasets to test the generalization of our findings:

MNIST: A widely-used image classification dataset [?], with 70,000 grayscale images of handwritten digits (0–9), each of size 28×28. The dataset is split into 60,000 training and 10,000 testing samples. It serves as representative case for high-dimensional, unstructured, visual data.

ISCAS85 and EPFL Circuit Netlists: These are standard benchmarks in the field of digital hardware design and verification [?, ?]. They contain combinational circuit representations in the form of netlists, which describe logic gates and their connections at the gate level.

For the ISCAS85 and EPFL datasets, we convert the netlists into ML-compatible tabular datasets using a self-developed netlist parser, which extracts structural features such as gate type, fan-in/fan-out, logic level, and connectivity patterns. This conversion pipeline is available in our open-source code release. The parser performs graph traversal to extract topological information (e.g., logic depth, gate fanout), applies one-hot encoding to gate types, and derives structural features such as interconnect count and signal direction. Unlike image or text datasets, the ISCAS85 and EPFL datasets present a unique challenge due to their graph-based, non-i.i.d. (independent and identically distributed) structural and categorical features, requiring careful preprocessing to map them into feature vectors suitable for HDC encoding. Those details are also available in our release.

All experiments were conducted on a Red Hat Enterprise Linux 8.6 server equipped with 128-core AMD EPYC 7763 processors, 1 TB of physical RAM, and a Python 3.10 environment. This setup enabled efficient parallel training and inference across multiple configurations and metrics, especially useful for evaluating large-scale hyper-

vector models and running grid searches specifically for MicroHD optimization (Section 4.7).

6 Experimental Results and Discussion

The HDC model for MNIST image and ISCAS85+EPFL circuit datasets has been trained as per the HDC architecture given in Section 2.2. Inference evaluation metrics and MIA metrics are shown in Table 2 marked as Baseline. We subsequently applied all architectural variants and enhancements detailed in Section 4, maintaining consistent feature representations and labels across all models to enable direct comparison. Membership inference analysis was performed uniformly across both baseline HDC models and their enhanced counterparts to quantify potential differences in privacy vulnerability. The value of hyperparameters and settings of experiments (corresponding to Section 4) is available in our source-code release.

6.1 Aggregate Observations from Experimental Results

Our comprehensive evaluation across MNIST and digital circuit datasets reveals fundamental trends in HDC's accuracy-privacy relationship:

– Accuracy Gains: Architectural enhancements consistently improve classification performance, with hybrid HDC models achieving the highest gains (96.31% MNIST, 78.05% circuits). Iterative retraining demonstrates particularly strong results across both domains (92.07% MNIST, 75.13% circuits), confirming the value of error-corrective learning in HDC.

– Privacy-Stability Trade-off: While accuracy improvements are substantial, most enhancements maintain or reduce membership leakage. Adaptive encoding and dimensionality optimization emerge as particularly balanced approaches, delivering 88-89% MNIST and 73-78% circuit accuracy while keeping cosine TPR below 40-45%—outperforming their baseline counterparts in both metrics.

– Generalization Benefits: Techniques that improve model generalization (iterative retraining, MicroHD) consistently show 5-15% lower MIA success rates compared to baseline, supporting the hypothesis that robust learning inherently mitigates membership memorization (Section 4). This effect holds across both image and circuit domains despite their structural differences.

– Dataset-Specific Patterns: MNIST models exhibit better privacy preservation (TPR mostly 36-46%), whereas circuit models display higher baseline vulnerability (TPR up to 73%), suggesting that the binary nature of circuit netlists may inherently increase membership traceability.

– Metric Consistency: Across all architectures, Hamming distance demonstrates the most stable TPR/TNR balance (47-57% MNIST, 28-59% circuits), while cosine similarity provides the clearest separation between high/low-risk models. Euclidean metrics show intermediate behavior, making them less reliable for comparative assessments.

6.2 Implications for HDC Security

Our results challenge the conventional wisdom that improved model utility necessarily compromises data privacy. Three key security insights emerge:

– Accuracy-Privacy Synergy: Contrary to trends in NNs, HDC demonstrates that accuracy improvements (e.g., +15% in hybrid models) can coincide with reduced membership leakage (5-15% lower TPR). This suggests HDC's transparent memory model may inherently resist the overfitting-privacy trade-off common in other paradigms.

– Architectural Leverage Points: The success of iterative retraining (36% TPR cosine) and adaptive encoding (39% TPR) confirms that data-aware training strategies can simultaneously enhance accuracy and obscure membership signals. This dual benefit stems from HDC's linear learning dynamics, where error correction naturally dilutes sample-specific traces.

– Domain-Dependent Vulnerabilities: The higher MIA success on circuit netlists (up to 73% TPR) versus MNIST (max 57% TPR) reveals that input data characteristics critically influence privacy risks. Binary circuit representations appear more susceptible to membership tracing than continuous image features, suggesting the need for domain-specific protections.

These findings validate threshold-based similarity analysis as an effective leakage detection framework for HDC, while highlighting that security guarantees must be evaluated relative to both model architecture and data modality. Future secure HDC designs should prioritize regularization via iterative learning over obfuscation, data-adaptive encoding over rigid projection schemes, and domain-tailored privacy budgets.

6.3 Comparison with Traditional ML Models

To contextualize the privacy characteristics of HDC models, we compare their performance and vulnerability to MIAs against a set of traditional ML models configured as follows:

– Fully Connected NN: 1 hidden layer (64 units), ReLU activation, and softmax output for MNIST images.

TABLE 2. HDC model accuracy and MIA metrics for MNIST and ISCAS85 + EPFL datasets using baseline HDC model (Section 2.2) as well as different modified architectures given in Section 4.

MNIST									
HDC Type	Dimension	Accuracy (%)		Cosine		Euclidean		Hamming	
		Train	Test	TPR (%)	TNR (%)	TPR (%)	TNR (%)	TPR (%)	TNR (%)
Baseline	10000	80.83	81.22	41	59	49	51	53	47
Iterative Retrain & Regeneration	40000	92.05	92.07	36	63	43	56	44	54
Adaptive Encoding	40000	88.21	88.62	39	60	48	56	43	55
Hybrid HDC	20000	97.39	96.31	46	52	54	45	57	42
Higher-Order Representation	40000	80.54	81.47	44	55	50	50	51	48
Dimensionality Optimization	20000	88.11	88.83	39	60	44	55	46	53
Error-Correcting Codes	40000	88.22	88.62	39	60	45	54	46	53
MicroHD Optimization	40000	88.45	89.27	38	61	44	55	46	53

ISCAS85 + EPFL									
HDC Type	Dimension	Accuracy (%)		Cosine		Euclidean		Hamming	
		Train	Test	TPR(%)	TNR (%)	TPR (%)	TNR (%)	TPR (%)	TNR (%)
Baseline	40000	70.96	70.57	53	47	44	56	57	43
Iterative Retrain & Regeneration	40000	75.65	75.13	50	50	42	59	55	45
Adaptive Encoding	40000	73.12	73.07	57	44	49	51	59	42
Hybrid HDC	40000	78.49	78.05	67	33	56	44	73	28
Higher-Order Representation	60000	76.35	72.84	45	54	48	51	49	50
Dimensionality Optimization	80000	77.35	74.23	41	58	44	55	45	54
Error-Correcting Codes	60000	77.65	73.21	42	57	45	54	45	54
MicroHD Optimization	50000	78.35	75.21	41	59	43	56	44	55

TABLE 3. Comparison of accuracy and MIA metrics between HDC and traditional ML models.

Model	Train Acc.	Test Acc.	Avg. TPR	Avg. TNR
MNIST				
HDC (Best Variant)	97.39%	96.31%	52.3%	46.3%
Fully Connected NN	99.11%	97.26%	0%	100.0%
DT	94.15%	90.85%	70.0%	30.0%
ISCAS85 + EPFL				
HDC (Best Variant)	78.49	78.05	65.33%	35%
Graph Convolution NN	92.49	92.48	67.97%	32.59%
DT	99.96	98.73	0	100%

– Graph Convolution NN: 1 hidden layer (32 nodes), ReLU activation, and softmax output for ISCAS85+EPFL netlist for combinational logic gate type classification.

– DT: A scikit-learn classifier with maximum depth tuned via grid search for both datasets.

The models were trained and evaluated under identical settings as in Section 6.1 using the MNIST and ISCAS85 + EPFL datasets. Table 3 presents the classification accuracy (train/test) and MIA vulnerability. For HDC, we compute the mean TPR and TNR from the membership results reported in Table 2.

The comparison reveals several significant insights regarding both utility and privacy characteristics of HDC relative to traditional models:

– HDC's Structural Vulnerability: HDC exhibits consistent membership leakage across domains, with 52.3% TPR for MNIST and 65.33% TPR for circuits, respectively. It demonstrates higher attack success than NNs in all cases (0% for MNIST and 32.59% for circuits), confirming the hypothesis that HDC's additive training mechanism preserves measurable training data traces.

– NNs' Inherent Robustness: NNs achieve 0% TPR on MNIST, indicating perfect privacy preservation, maintain strong protection on circuits (32.59% TPR), and suggest that non-linear transformations effectively obscure membership signals.

– DTs' Polar Behavior: DTs exhibit 70% TPR on MNIST, suggesting vulnerability, while achieving 0% TPR on circuits, indicating perfect privacy. This contrast highlights the extreme domain-dependence in privacy characteristics.

7 Proposed Defense Strategies

Building on our findings about HDC's vulnerability to MIAs, we present a defense framework that systematically addresses three attack surfaces through architectural enhancements.

7.1 Unified Defense Framework

Our analysis reveals that HDC's vulnerability to membership inference stems from three core weaknesses: transparent feature encoding, sample-specific memorization, and deterministic similarity scoring. To address these, we illustrate a unified defense framework through Table 1 that consolidates the strengths of seven strategically tuned HDC configurations in Section 4. While each of those configurations was initially designed to improve HDC model accuracy, we observe that they collectively introduce a variety of properties—such as generalization, obfuscation, redundancy, and robustness—that are inherently conducive to defending against MIAs.

7.2 Representation Obfuscation

The first line of defense disrupts the direct mapping between inputs and model internals. Adaptive encoding techniques (Section 4.2) replace random projections with data-aware transformations, using PCA to inject controlled entropy into feature representations. Higher-order binding operations (Section 4.4) further obscure sample-specific patterns through multiplicative feature interactions. Our experiments show these techniques reduce membership inference success rates by 25.3% while incurring only a 1.6% accuracy penalty on MNIST classification tasks as shown in Table 4.

7.3 Robust Generalization

The results in Section 6 show we can mitigate overfitting through iterative retraining (Section 4.1) and hybrid architectures (Section 4.3). The retraining process dynamically adjusts class hypervectors to focus on challenging samples, smoothing decision boundaries without compromising model capacity. Hybrid models combine neural feature extractors with HDC classification, maintaining 96.31% accuracy while limiting TPR increases to under 5%. These approaches demonstrate that improved generalization inherently mitigates membership leakage, as evidenced by the 31.2% reduction in attack success rates.

7.4 Error-Resilient Decisions

Final layer protections introduce redundancy through ECCs (Section 4.6) and optimized dimensionality (Section 4.5). By encoding class labels with 15-20% redundancy, the model becomes less sensitive to membership probes while preserving classification reliability. Dimensionality optimization tailors the hypervector space to the needs of each dataset, achieving 41% TPR at 74. 23% precision for circuit netlists—a favorable trade-off compared to the baseline vulnerability of 65. 33%.

TABLE 4. Defense impact on MIA vulnerability calculated from Table 2.

Strategy	TPR Reduction	Accuracy Impact
Adaptive Encoding	25.3%	-1.6%
Iterative Retraining	31.2%	+0.2%
Error-Correcting Codes	25.4%	-0.8%

This coordinated approach demonstrates that architectural modifications can simultaneously enhance accuracy and privacy, a critical advantage for edge deployments where computational resources are limited. Future work will formalize theoretical privacy bounds and extend these principles to thwart model inversion attacks while maintaining HDC's energy efficiency.

8 Conclusion

This work demonstrates HDC models' inherent MIA vulnerability, with 65.33% success on circuit netlists and 52.3% on MNIST—surpassing traditional ML models. Architectural analysis reveals HDC's additive training and transparent representation directly enable privacy risks, while defenses like adaptive encoding achieve 25-31% effectiveness reductions. The findings challenge HDC's presumed security advantages, particularly for hardware security and edge AI. We provide the first formal framework for evaluating and mitigating HDC privacy risks, establishing critical baselines for secure hyperdimensional computing designs.