Proceedings of the ASME 2023
International Design Engineering Technical Conferences and
Computers and Information in Engineering Conference
IDETC-CIE2023
August 20-23, 2023, Boston, Massachusetts

DETC2023-110699

HYPERDIMENSIONAL COGNITIVE COMPUTING FOR LIGHTWEIGHT CYBERATTACK DETECTION IN INDUSTRIAL INTERNET OF THINGS

Fardin Jalil Piran[1], Hamza Errahmouni Barkam[2], Mohsen Imani[2], Farhad Imani[1,*]

[1]Department of Mechanical Engineering, University of Connecticut, Storrs, CT
[2]Department of Computer Science, University of California Irvine, Irvine, CA

ABSTRACT

Although the connectivity offered by industrial internet of things (IIoT) enables enhanced operational capabilities, the exposure of systems to significant cybersecurity risks poses critical challenges. Recently, machine learning (ML) algorithms such as feature-based support vector machines and logistic regression, together with end-to-end deep neural networks, have been implemented to detect intrusions, including command injection, denial of service, reconnaissance, and backdoor attacks, by capturing anomalous patterns. However, ML algorithms not only fall short in agile identification of intrusion with few samples, but also fail in adapting to new data or environments. This paper introduces hyperdimensional computing (HDC) as a new cognitive computing paradigm that mimics brain functionality to detect intrusions in IIoT systems. HDC encodes real-time data into a high-dimensional representation, allowing for ultra-efficient learning and analysis with limited samples and a few passes. Additionally, we incorporate the concept of regenerating brain cells into hyperdimensional computing to further improve learning capability and reduce the required memory. Experimental results on the WUSTL-IIOT-2021 dataset show that HDC detects intrusion with the accuracy of 92.6%*, which is superior to multi-layer perceptron (*40.2%*), support vector machine (*72.9%*), logistic regression (*84.2%*), and Gaussian process classification (*89.1%*) while requires only 300 data and 5 iterations for training.*

Keywords: Industrial internet of things, Cyberattack detection, Cognitive computing

1. INTRODUCTION

The recent convergence of low-cost sensing, communication, and computation technologies has presented an unprecedented opportunity for the widespread adoption of the industrial internet of things (IIoT). To provide a comprehensive view of operations, a large number of field devices (e.g., actuators and sensors), machines, robots, gateways, and data acquisition systems are connected to IIoT network. The connectivity is facilitated through wireless communication protocols, including through wireless communication protocols constrained application protocol (CoAP), distributed network protocol version 3 (DNP3), message queuing telemetry transport (MQTT) protocol, and Modbus [1, 2]. The seamless connectivity and data sharing in IIoT not only transform inventory and supply chain optimization but also enable predictive maintenance, fault diagnosis, machine monitoring, and operator safety, to name a few [3].

However, the extensive communication among devices in IIoT and the vulnerabilities present in information networks and cloud databases introduce significant cybersecurity risks, which affect various components such as remote terminal units, human-computer interfaces, communication infrastructure, application servers, and even actuators and sensors. In particular, cyber intrusions leveraging vulnerabilities in communication protocols result in different types of attacks, such as denial of service, command injection, reconnaissance, and backdoor [4–6]. By inserting malicious code into vulnerable systems, command injection attacks execute unauthorized rulings to compromise the availability, confidentiality, and integrity of IIoT. For example, structured query language (SQL) as an injection attack seeks to compromise or control the database servers. The attack allows an intruder to manipulate control commands in the system, disrupting normal operations [7]. Denial of service (DoS) attacks overload a system with traffic to disrupt operations or cause system failures, which result in equipment damage and production delays [8]. Reconnaissance attacks determine weaknesses and potential attack vectors to launch targeted attacks. Here, attackers leverage sniffers to eavesdrop on ongoing network traffic, allowing gathering information about network components and current status in silent [9]. Backdoor attacks permit continuous access to systems to disrupt operations, steal data, or carry out further attacks [10].

*Corresponding author: farhad.imani@uconn.edu

To mitigate intrusion risks, various security measures are executed, including encryption approaches, strict access controls, network segmentation models, as well as intrusion detections and prevention mechanisms [11, 12]. Data encryption plays a crucial role in safeguarding sensitive information within IIoT systems. By encoding data into a format that is readable only by authorized users, encryption helps protect valuable data such as device credentials, sensor data, and configuration files from unauthorized access. Several common approaches in this category include the triple data encryption algorithm, elliptic curve digital signature algorithm, and asymmetric cryptography [13]. In addition to encryption, many industries are adopting zero-trust security models and implementing secure coding practices to further enhance the security of IIoT systems. Zero-trust models operate under the assumption that all devices and users are untrustworthy until proven otherwise. These models require continuous authentication and authorization processes to control access to resources, reducing the attack surface of IIoT systems and minimizing the risk of unauthorized access [14, 15].

However, the utilization of cryptographic solutions in IIoT systems introduces certain drawbacks. First, the solutions increase the processing and bandwidth requirements of the system, as additional computation and data overhead are necessary for data encryption and decryption. Consequently, this leads to slower system performance and increased resource consumption, potentially impacting the overall efficiency of IIoT operations [16, 17]. Second, implementing and managing data encryption at scale is challenging. Encryption keys not only need to be securely generated and managed but also require an effective mechanism for distributing keys to authorized users. This adds complexity to the system and necessitates robust key management practices to ensure the secure sharing and revocation of keys. Third, the compromise or loss of encryption keys results in the permanent detriment of encrypted data, as there is no method to recover the data without the original key [18, 19]. Finally, some legacy field devices and equipment in industrial systems are not designed to withstand current threat vectors or receive cryptographic solutions. The comprehensive review of encryption challenges in IIoT can be found in [20].

Cyberattack monitoring is another strategy in IIoT security, which involves continuously observing systems for signs of intrusions. Monitoring is performed through a range of techniques, including intrusion detection and prevention systems, log analysis, and threat intelligence feeds. Huma et al. [21] introduced a deep learning-based system that utilizes a novel random hybrid neural network (HDRaNN) to detect intrusion in IIoT. Latif et al. [22] designed a deep learning model based on a random neural network to detect various cyberattacks. The model was validated using a DS2OS dataset that contained 357,952 samples with seven different types of attacks. Trane et al. [23] applied random forest to detect cyberattacks. Abdullahi et al. [24] established a thorough study on the capability of different ML models for cyberattack detection in IoT and Industry 4.0 environments. According to the authors, support vector machines (SVM) was one of the most frequently employed techniques due to exceptional accuracy in detecting intrusions. Nevertheless, the authors highlighted other high-performance methods, such as neural networks (NN) and recurrent neural networks (RNN). However, the challenges related to monitoring IIoT systems are as follows:

1. **Dynamic attack vectors:** Ongoing emergence of novel attack vectors, such as communication channels and denial-of-service attacks, pose significant challenges to ML models for intrusion detection. Monitoring can be ineffective against new or unknown types of cyberattacks, as the attack may not match any known patterns or signatures [21, 25].

2. **Data limitation:** Many industries are hesitant to report cyberattacks due to the potential damage to reputation or leakage of confidential information. The lack of accessing appropriate data makes evaluating the security risks and threats difficult. Thus, designing algorithms that can learn with very few samples is essential.

3. **Human-like cognitive support:** Although machine learning algorithms are strong in finding anomalies and irregularities, they are still far short of capabilities to replace human intelligence to understand, analyze, and reason a cyberattack. To enable reliable and informative prediction, cybersecurity applications need to support human-like cognition to provide a higher learning quality and to reason for each prediction or decision [26–28].

We introduce hyperdimensional computing (HDC) as a new cognitive computing paradigm capable of detecting cyberattacks in industrial internet of things systems with sparse data in a few passes [26–34]. HDC leverages high-dimensional vectors to represent data and operations. It is based on the theory that the brain stores and processes information using high-dimensional vectors rather than the binary representations used in traditional computing. HDC has several advantages over traditional computing approaches for cyberattack detection in IIoT systems. First, HDC identifies attacks in the hypervectors used to represent the data, making it particularly effective in detecting new and unknown types of cyberattacks that may not match any known patterns or signatures. Second, HDC is an online model that continuously updates hypervectors based on new data at high speeds. This enables industries to detect cyberattacks quickly and respond in real time, reducing the impact of attacks on their operations. Third, HDC, as an intuitive and human-interpretable framework, exposes hidden features, enabling single-pass or few-passes learning with just a few samples. These make HDC a powerful tool for lightweight, online, and cognitive cyberattack detection in IIoT systems that continually learns and adapts to new types of attacks, improving accuracy and effectiveness over time.

The structure of this paper is as follows: The proposed methodology is presented in Section 2. Section 3 describes the experimental design. The outcomes of a case study are presented in Section 4. Finally, Section 5 summarizes the shortcomings of current learning methods for IIoT cyberattack detection and outlines future research directions.

2. METHODOLOGY

As shown in Figure 1, the HDC framework is based on the abstract representation of neuronal circuits in the human brain,

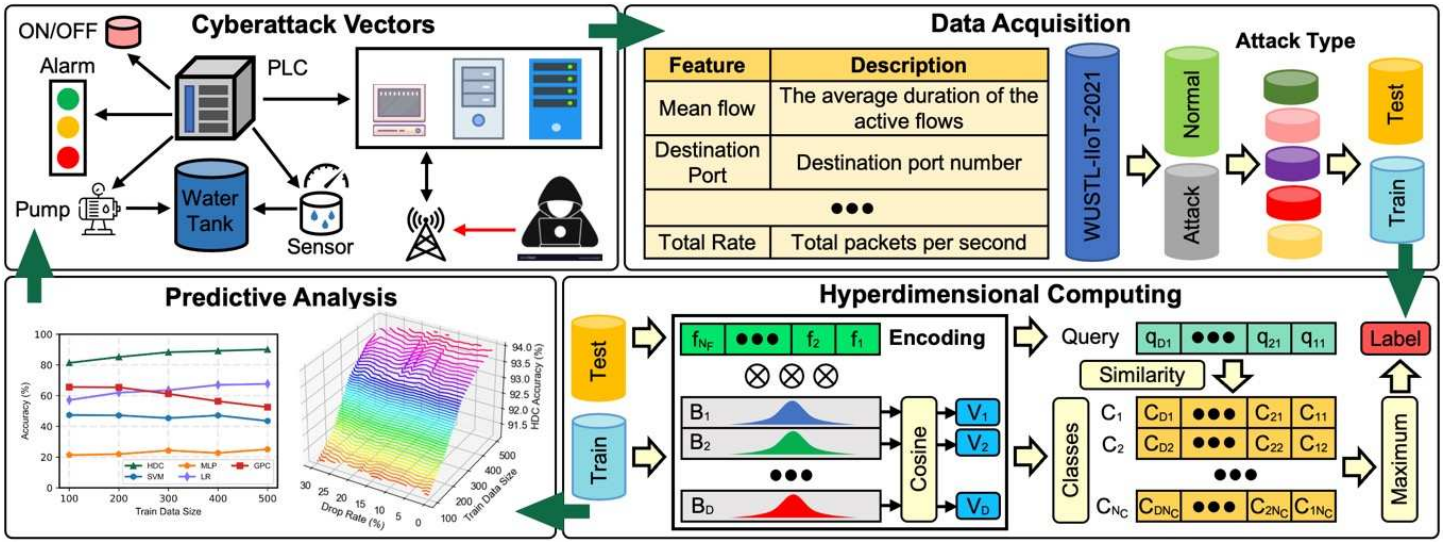

FIGURE 1: FLOWCHART OF THE HYPERDIMENSIONAL COMPUTING FOR CYBERATTACK DETECTIONS IN IIOT.

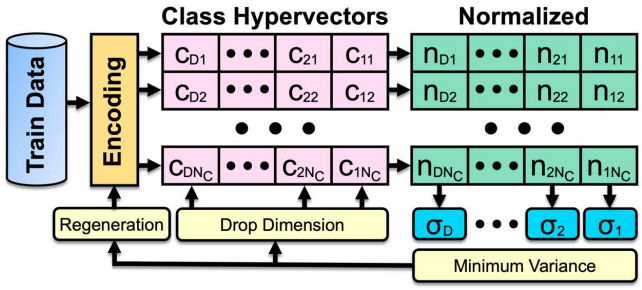

FIGURE 2: OVERVIEW OF HDC TRAINING AND REGENERATION IN CYBERATTACK MONITORING.

and it involves three stages: encoding, online training, and iterative learning. Initially, the HDC encodes the training data into the high-dimensional space. Next, it generates hypervectors for each class, which are used for single-pass training over the encoded data. HDC then compares the similarity between class hypervectors and encoded query data and retrains from assigning the closest class to the query. This iterative learning process allows HDC to continuously improve accuracy in intrusion detection.

2.1 HDC Encoding

The first step in hyperdimensional computing is to encode the input data into hypervectors in a high-dimensional space. The hypervectors contain all the information across all components, with no component being more responsible for storing any piece of information than another. HDC employs different encoding methods depending on the data type. Let's consider a function that encodes a feature vector $\vec{F} = \{f_i\}_{i=1}^{N_F}$ with $f_i \in \mathbb{R}$ into a hypervector $\vec{V}_i$. Suppose $\vec{V}_i \, \forall i = 1 : N_T$ are randomly generated hypervectors ($\vec{V}_i \in [-1, +1]^D$). Each dimension of the encoded data is generated by taking the dot product of the feature vector with a randomly generated vector, resulting in

$$v_i = \cos(\vec{F} \cdot \vec{B}_i) \tag{1}$$

where $\vec{B}_i$ is a randomly generated vector from a Gaussian distribution with a mean of 0 and a standard deviation of 1. The set of random vectors $\{\vec{B}_i\}_{i=1}^{D}$ can be generated offline and then used for the entire cyberattack detection task (where $\vec{B}_i \in \mathbb{R}^{N_F}$). After this encoding step, each element v_i represents an element of the hypervector $\vec{V}$. These hypervectors are nearly orthogonal, meaning that $\delta(\vec{V}_i, \vec{V}_j) \approx 0 \, \forall i, j = 1 : N_T \, \& i \neq j$, where δ denotes the cosine similarity. HDC uses the following set of primitives for encoding cyberattack data:

Bundling (+): This is an element-wise addition of multiple hypervectors. Bundling generates another hypervector with the same dimensionality as inputs. In high-dimensional space, bundling is similar to a memory operation, where the bundled hypervector remembers the input operands' information.

Binding ($*$): This operation associates the information of multiple objects into a single hypervector. Binding is defined as a bitwise XOR operation in binary, and multiplication operation in the bipolar domain ($\vec{V} = \vec{V}_i * \vec{V}_j$). The binded hypervector is a new object in high-dimensional space that is orthogonal to all input hypervectors.

2.2 HDC Regeneration

We introduce a new approach to HDC, which utilizes a dynamic encoder for adaptive learning. In HDC, not all dimensions have the same impact on the learning task, with some dimensions having little or no effect. The aim of HDC is to identify these insignificant dimensions and remove them from the computation. To improve accuracy, HDC regenerates these dimensions in the encoding module, giving them a new opportunity to contribute more significantly to the learning task.

Figure 2 provides an overview of the HDC learning framework. Initially, HDC maps the data into high-dimensional space using one of the existing encoding methods. The encoding is dependent on the data type and performed using defined HDC mathematics over a set of randomly generated base vectors (e.g., as random hypervectors in $D = 10,000$ dimensions).

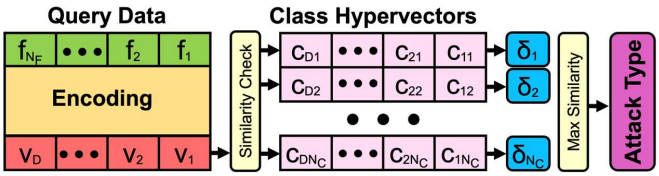

FIGURE 3: OVERVIEW OF HDC INFERENCE IN CYBERATTACK MONITORING.

The HDC model is trained over the training data and then normalized to simplify the similarity metric for a dot product operation in inference or retraining. Next, we compute the variance throughout all elements of class hypervectors with minimum variance as they represent the dimensions that can be regenerated. The HDC drops the dimensions from the framework and base hypervectors in the encoder part. The framework regenerates the base hypervectors on the indicated dimensions. To ensure continued learning, HDC employs two iterative learning methods that involve repeating training, dimension reduction, and dimension regeneration. This process continues until HDC identifies a model where the majority of dimensions contribute significantly to the classification. Additionally, HDC makes use of compressed and efficient representations, which can reduce the computational overhead and memory requirements of the learning process.

Drop Dimension: During the training phase of hyperdimensional computing, a single hypervector representing each class is created. The inference task is performed by checking the similarity between a query hypervector (encoded inference data) and all class hypervectors. The query data is then assigned to the class with the highest similarity. The objective of HDC is to train class hypervectors, where their patterns represent information. A weak classifier cannot identify distinct patterns for different classes, resulting in multiple classes having similar patterns. This makes the classification task challenging, as the query may have a high similarity value to multiple classes.

Figure 3 illustrates how HDC calculates similarity between a query, $\vec{V} = \{v_i\}_{i=1}^{D}$, and class hypervectors. During the learning phase, HDC computes the cosine similarity as:

$$\delta(\vec{C_j}, \vec{V_i}) = \frac{\vec{C_j} \cdot \vec{V_i}}{\|\vec{C_j}\| \cdot \|\vec{V_i}\|} \tag{2}$$

where the nominator is the inner product between the class hypervector $\vec{C_j}$ and the query $\vec{V_i}$. The closer the δ is to 1, the more similarity exists between two hypervectors. Our objective is to identify dimensions that have little impact on the classification task. Therefore, we compute the variance over each dimension of the classes. Dimensions with low variance have similar values across all classes, indicating that they store common information. During the search, these dimensions add similar weights to cosine over all classes. We refer to these dimensions as insignificant since they do not contribute significantly to differentiating the class patterns.

Dimension Regeneration: At the beginning of the training phase, HDC creates an initial HDC model, which consists of a hypervector representing each class. For each dimension of the class hypervector, HDC computes the variance over the normalized model. Next, HDC selects a percentage of dimensions with the minimum variance as candidates for dropping. Instead of leaving these dropped dimensions blank, HDC regenerates them. The primary objective of regeneration is to create new dimensions that can potentially have a greater impact on classification, providing a higher variance. By continually dropping insignificant dimensions and regenerating them during the training phase, HDC aims to identify the most influential dimensions for classification, which improves accuracy. Overall, this approach offers a promising solution for hyperdimensional computing in various applications such as text classification and image recognition.

2.3 HDC Retraining

HDC employs a continual learning approach, building on the previously learned model instead of starting from scratch. During the training phase, only the values of the dropped dimensions are ignored, while other dimensions continue to learn based on their existing values. For each input data, the similarity between the hypervector $\{\vec{V_i}\}_{i=1}^{N_T}$ and all class hypervectors $\vec{C}$ is checked. The model is updated for all mispredicted cases. If the model predicts input as cyberattack type $j^{'}$ instead of j, the retraining updates the model using the update rate β. This process ensures the model updates how far a training query is mispredicted. If the similarity value of $\delta(\vec{C}j^{'}, \vec{V_i})$ is considerably higher than $\delta(\vec{C_j}, \vec{V_i})$, it indicates that the prediction result is significantly different from the actual result. In this case, the retraining step makes significant changes to the model. However, if $\delta(\vec{C_j}, \vec{V_i}) \simeq \delta(\vec{C_j^{'}}, \vec{V_i})$, the retraining step only adds a small portion for the update. This process is applied to the entire dataset or a batch of data to generate a new model, which can be used for the inference task or another iteration of retraining.

3. EXPERIMENTAL DESIGN

In terms of monitoring industrial processes and machines, industrial control systems (ICSs) have been used for a long time as part of critical infrastructures. They consist mainly of supervisory control and data acquisition systems that have a graphical interface through its human-machine interface (HMI). The IIoT system, shown in Figure 4, monitors the water level and turbidity quantity of the water storage tank. A water treatment and distribution system like this is installed in industrial reservoirs to treat and distribute water. Among the components of this testbed are historical logs, HMIs, and Programmable Logic Controllers (PLCs). This testbed has three sensors and four actuators. An analog turbidity sensor and two water level sensors make up the inputs. PLC commands are received by a three-light turbidity alarm, a valve, and two water pumps. There are also control buttons (on, off, and light indicators) for manual operation. The purpose of this testbed is to maintain the water level between two predefined levels. Additionally, it measures the degree of turbidity in the water and illuminates either a red, yellow, or green light of the turbidity alarm, depending on the level of cloudiness. A popular IIoT protocol commonly used in ICSs is Modbus, which was used in this testbed as the communication protocol. Ladder language is implemented to program the logic of the PLC.

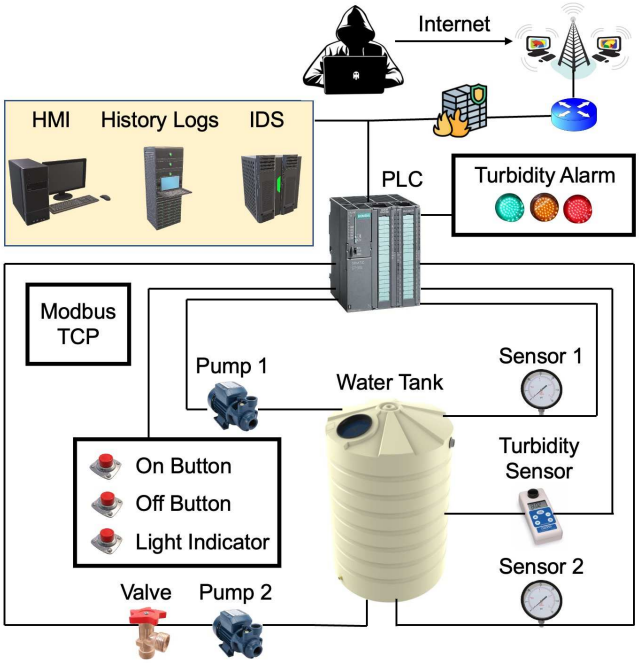

FIGURE 4: OVERVIEW OF THE IIOT STRUCTURE THAT IS UTILIZED TO CREATE EXPERIMENTAL DATA.

ICSs include supervisory control and data acquisition systems with human-machine interfaces, which are traditionally deployed as standalone systems with minimal external connectivity. However, with the integration of IIoT technology and the increased use of internet communications, ICSs have become more exposed to potential security threats. Feature extraction from traffic is essential for developing a dataset for cyberattack detection. In designing the intrusion detection system (IDS), they select the features whose values differed between phases of attack and normal operation. Unless a selected feature varies during an attack, no algorithm can detect an intrusion or anomalous condition using that feature. Through the use of the Argus tool [35], the potential features are analyzed, and 41 of them that are selected based on prevalent network flows and possible change during the attack phase. Source port, mean flow, total percent loss, and destination port are examples of the features in this dataset. Total percent loss is the percentage of packets dropped or re-transmitted, and destination port is the destination port number. The source port is the port number of the source, and the mean flow is the average duration of active flows [25].

The WUSTL-IIoT-2021 dataset provides a valuable resource for conducting cybersecurity studies on IIoT networks. It includes real-world industrial systems with real-time cyberattacks, consisting of normal traffic and four types of attack traffic: Command Injection, DoS, Reconnaissance, and Backdoor [25]. To remove potentially identifying features, we excluded StartTime, LastTime, SrcAddr, DstAddr, sIpId, and dIpId. The remaining dataset contains 41 features, which are linearly mapped to a range of zero to one. Given the focus on edge computing, the classifier is trained on a small subset of the data.

As the dataset is unbalanced with the majority of samples being normal traffic, we balanced the sample data into equal

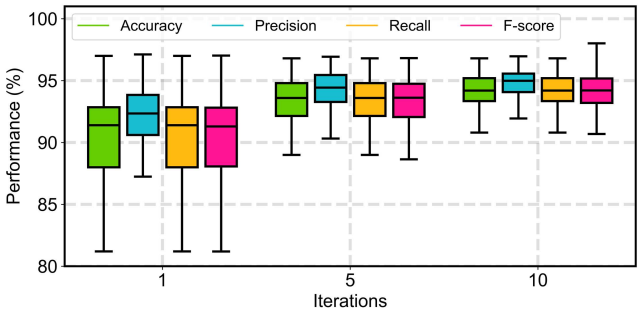

FIGURE 5: THE PERFORMANCE METRICS OF HDC CORRESPOND TO THE DIFFERENT NUMBERS OF ITERATIONS.

groups for each class. The Backdoor attack has the smallest sample size of 212, which sets the minimum sample size across all classes. We performed a classification task with varying training data sizes, selecting 20, 40, 60, 80, and 100 samples randomly from each class as training data. For evaluation, we randomly selected the same number of samples from each class as test data. This approach ensures equal train and test data sizes, although selecting the data is a challenging task.

To evaluate classifier performance on the test data, we used four evaluation metrics: accuracy, precision, recall, and F-score. We repeated the process of selecting train and test data 100 times and calculated the four evaluation metrics for each iteration. Finally, we reported the average and standard deviation (STD) of the evaluation metrics over the 100 repetitions.

4. EXPERIMENTAL RESULTS

4.1 Hyperdimensional Computing Results

The HDC can be trained in just a few iterations, but it's important to evaluate its performance over multiple iterations. For our experiment, we used 100 samples as training data and 100 samples as test data for each class. We set the hypervector dimension to 2000 and the learning rate to 0.037, as per the basic HDC model. Throughout the experimental section, these parameters remained fixed.

Figure 5 shows the evaluation of the HDC over multiple iterations. The accuracy, precision, recall, and F-score for the first iteration (one-pass learning) were 89.1%, 90.5%, 89.1%, and 88.7%, respectively. This demonstrates that the HDC is fast, achieving around 90% performance with just one pass of the data and no retraining. With increased iterations, the average of the evaluation metrics increased while the standard deviation (STD) decreased. For instance, with five iterations, the accuracy, precision, recall, and F-score were 92.6%, 93.4%, 92.6%, and 92.4%, respectively. With ten iterations, the performance further increased to 93.9%, 94.6%, 93.9%, and 93.8%, respectively. This represents a 4% improvement, demonstrating the HDC's ability to improve performance quickly. After 20 iterations, the accuracy, precision, recall, and F-score reached 94.8%, 95.3%, 94.8%, and 94.8%, respectively. However, the improvement from ten to twenty iterations was only 1%, suggesting that the HDC reaches a steady state after ten iterations. Based on the STDs for ten iterations, the metrics values were consistently

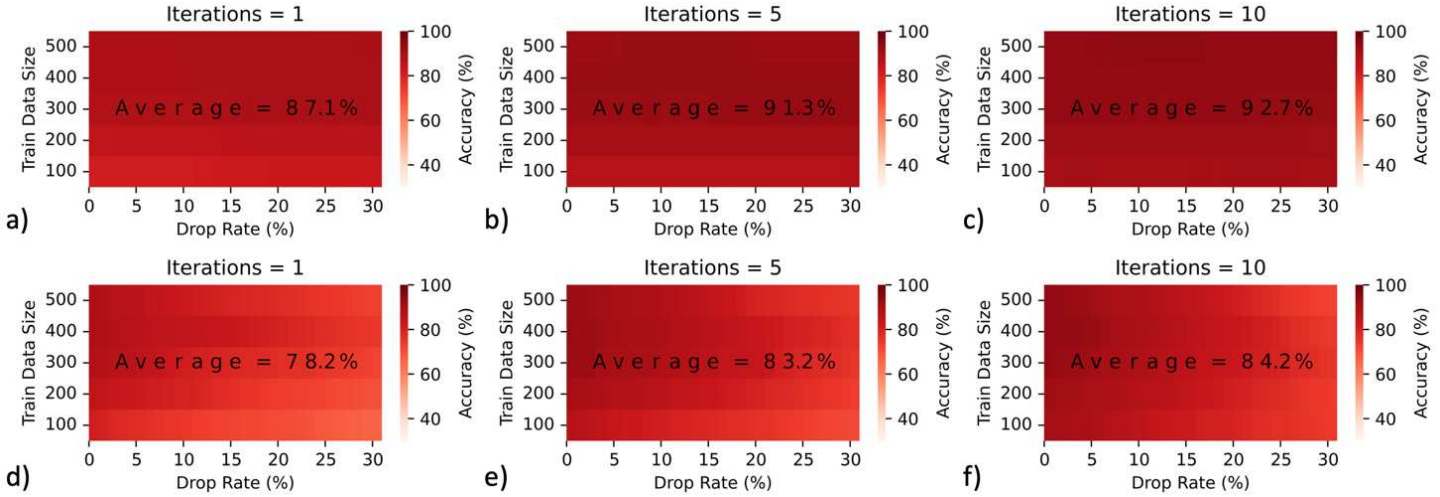

FIGURE 6: THE ACCURACY OF HDC OVER DROP RATE AND TRAIN DATA SIZE FOR A) 1 ITERATION AND B) 5 ITERATIONS AND C) 10 ITERATIONS WHEN DIMENSIONS WITH THE LOW VARIANCE OF CLASS HYPERVECTORS ARE SELECTED FOR DROPPING. THE FIGURES D)-F) REPRESENTS THE ACCURACY OVER, 1, 5, AND 10 ITERATIONS, RESPECTIVELY, WHEN HIGH-VARIANCE ELEMENTS OF CLASS HYPERVECTORS ARE CHOSEN FOR DROPPING.

above 90%. Therefore, the HDC can be trained effectively with just ten iterations to achieve reliable performance.

Figure 6 illustrates the effect of dropping hypervectors dimensions on HDC performance over train data size using six heatmaps. In a heatmap, the y-axis, and the x-axis show the train data size and drop rate percentage. There is a range of train data sizes from 100 to 500 and a drop rate ranging from zero to thirty percent. The heatmaps show the accuracy and the average accuracy of each heatmap written. Figure 6 a), b), and c) show the effects of the drop dimensions with low variance when the HDC trained for one-pass learning, 5 and 10 iterations, respectively. If we do not drop dimensions, the average accuracy over the train data size is 86.5%, 91.2%, and 92.8%, respectively, for one, 5, and 10 iterations. The heatmap averages are 87.1%, 91.3%, and 92.7%, respectively, for one, 5, and 10 iterations which denote that the accuracy is almost unaffected by dropping low variance dimensions after 5 and 10 iterations. However, the accuracy is increased by dropping the low variance in one-pass learning since, in this case, we only map the data to high dimensional space, and if we regenerate the low variance dimensions, discrimination of the classes will be done better. For example, if we drop low variance dimensions up to 30% for 5 and 10 iterations and the train data set of 300, the accuracy changes by less than 0.3%. Nevertheless, in this instance, the accuracy of one-pass learning is increased by increasing the drop rate, and the accuracy goes from 88.2% to 89.8%. When dimensions are not dropped, the accuracy for 100, 200, 300, 400, and 500 train data sizes is 87.4%, 90.9%, 92.6%, 92.8%, and 92.6% for 5 iterations, as well as 91.2%, 92%, 93.2%, 93.7%, and 93.9% for 10 iterations. If the 20% of low variance dimensions are dropped, the accuracy is 87.4%, 91%, 92.8%, 92.9%, and 92.5% for five iterations, as well as 91.3%, 91.9%, 93.1%, 93.5%, and 93.9% for ten iterations. The accuracy has changed by equal and less than 0.2% over 5 iterations and 0.2% over 10 iterations, respectively, over train data sizes that show that dropping low variance dimensions does not affect HDC performance. With one-pass learning, the accuracy, for 100, 200, 300, 400, and 500 train data sizes, respectively, is 81.3%, 85.1%, 88.2%, 89.1%, and 89.1%, while after dropping 20% of low variance dimensions, the accuracy increases to 82.3%, 86.2%, 89.3%, 89.5%, and 89.8%. These results demonstrate that HDC performance can be improved by dropping low variance dimensions in one-pass learning. As a result, it can be concluded that the low variance dimensions have little impact on accuracy and are insignificant.

Figures 6 d), e), and f) show the effects of the drop dimensions with high variance when the HDC is trained for 1, 5, and 10 iterations. The heatmap average for dropping high variance dimensions is 78.2%, 83.2%, and 84.2% for 1, 5, and 10 iterations which means dropping high variance dimensions decreases the accuracy 8.3%, 8%, and 8.6%, respectively. In the study of a fixed train data size, increasing the drop rate decreases the accuracy of 1, 5, and 10 iterations for train data size 300. With one-pass learning, the accuracy is 88.2%, 85.2%, 80.9%, 78.4%, 76.7%, 74.8%, and 72.1% for 0%, 5%, 10%, 15%, 20%, 25%, and 30% drop rate. This case has an accuracy of 92.6%, 90.1%, 88.6%, 85.7%, 83.3%, 78.4%, and 75.4% for 5 iterations, as well as 93.2%, 91.4%, 88.9%, 85.8%, 82.2%, 79.2%, and 75.4% for 10 iterations. According to the results, increasing the drop rate of high variance dimensions reduces the accuracy for a fixed train data size and 1, 5, and 10 iterations. If the 20% of high variance dimensions are dropped for train data sizes of 100, 200, 300, 400, and 500, the accuracy is 70.7%, 73.5%, 76.7%, 80%, and 77.5% for one-pass learning, and 76.1%, 79.9%, 83.3%, 83.2%, and 79.6% for five iterations, as well as 79.4%, 80.3%, 82.2%, 83%, and 80.8% for ten iterations. The effectiveness of HDC in identifying insignificant dimensions and removing them from the computation, as well as regenerating them to improve accuracy, makes it a promising framework for hyperdimensional computing in IIoT systems.

TABLE 1: THE COMPARISON RESULTS OF HDC WITH SVM, MLP, LR, AND GPC ACCORDING TO ACCURACY, PRECISION, RECALL, AND F-SCORE OVER 1, 5, 10 ITERATIONS, AND TRAINING DATA SIZE OF 100 TO 500.

		Accuracy (%)			Precision (%)			Recall (%)			F-score (%)		
	Train Data Size	1	5	10	1	5	10	1	5	10	1	5	10
HDC	100	**81.3 ± 8.3**	87.4 ± 6.9	91.2 ± 3.4	**84.8 ± 8.1**	89.6 ± 6.7	92.6 ± 2.6	**81.3 ± 8.3**	87.4 ± 6.9	91.2 ± 3.4	**80.5 ± 9.4**	87.0 ± 7.7	91.0 ± 3.5
	200	85.1 ± 8.5	90.9 ± 5.4	92.0 ± 4.4	87.1 ± 8.9	91.9 ± 6.0	93.1 ± 4.5	85.1 ± 8.5	90.9 ± 5.4	92.0 ± 4.4	84.6 ± 9.7	90.7 ± 6.4	91.9 ± 5.1
	300	88.2 ± 5.7	**92.6 ± 2.6**	93.2 ± 2.7	89.7 ± 6.9	**93.7 ± 1.8**	94.1 ± 3.3	88.2 ± 5.7	**92.6 ± 2.6**	93.2 ± 2.7	87.8 ± 6.9	**92.6 ± 2.5**	93.1 ± 3.2
	400	89.1 ± 5.8	92.8 ± 3.2	93.7 ± 3.0	90.8 ± 5.9	93.8 ± 3.1	94.4 ± 3.0	89.1 ± 5.8	92.8 ± 3.2	93.7 ± 3.0	88.9 ± 6.5	92.8 ± 3.6	93.7 ± 3.3
	500	89.1 ± 6.4	92.6 ± 4.4	**93.9 ± 2.9**	90.5 ± 6.6	93.4 ± 4.9	**94.6 ± 2.3**	89.1 ± 6.4	92.6 ± 4.4	**93.9 ± 2.9**	88.7 ± 7.5	92.4 ± 5.1	**93.8 ± 3.2**
SVM	100	**47.3 ± 14.7**	82.6 ± 8.0	86.2 ± 3.8	**52.2 ± 18.2**	82.9 ± 10.7	88.6 ± 2.7	**47.3 ± 14.7**	82.6 ± 8.0	86.2 ± 3.8	**44.8 ± 16.7**	81.1 ± 9.8	85.9 ± 4.0
	200	47.1 ± 14.7	76.5 ± 8.0	85.7 ± 7.2	53.5 ± 16.5	75.8 ± 9.9	86.6 ± 8.0	47.1 ± 14.7	76.5 ± 8.0	85.7 ± 7.2	45.0 ± 16.6	74.2 ± 9.4	84.7 ± 8.6
	300	45.3 ± 12.1	**72.9 ± 7.9**	82.7 ± 7.5	51.7 ± 14.6	**73.9 ± 9.7**	82.5 ± 8.8	45.3 ± 12.1	**72.9 ± 7.9**	82.7 ± 7.5	42.9 ± 13.8	**71.1 ± 9.5**	81.0 ± 8.9
	400	47.1 ± 12.1	71.5 ± 8.5	82.7 ± 7.6	53.6 ± 15.1	73.2 ± 10.0	82.5 ± 9.0	47.1 ± 12.1	71.5 ± 8.5	82.7 ± 7.6	44.4 ± 13.7	69.8 ± 10.1	80.9 ± 9.2
	500	43.5 ± 12.7	69.8 ± 9.4	**80.9 ± 7.3**	49.4 ± 15.7	71.2 ± 10.4	**81.0 ± 8.3**	43.5 ± 12.7	69.8 ± 9.4	**80.9 ± 7.3**	40.6 ± 14.4	68.3 ± 10.8	**79.0 ± 8.9**
MLP	100	**21.3 ± 8.7**	29.6 ± 10.2	42.1 ± 12.4	**13.0 ± 10.9**	21.2 ± 14.2	38.8 ± 18.9	**21.3 ± 8.7**	29.6 ± 10.2	42.1 ± 12.4	**12.2 ± 7.7**	20.7 ± 11.2	34.8 ± 14.7
	200	21.9 ± 8.3	30.4 ± 10.4	44.1 ± 13.8	12.2 ± 9.5	23.3 ± 15.5	40.7 ± 18.0	21.9 ± 8.3	30.4 ± 10.4	44.1 ± 13.8	12.8 ± 7.1	20.5 ± 11.6	36.8 ± 16.2
	300	24.2 ± 9.6	**40.2 ± 10.9**	63.5 ± 11.9	15.1 ± 11.0	**36.8 ± 17.4**	66.6 ± 15.0	24.2 ± 9.6	**40.2 ± 10.9**	63.5 ± 11.9	15.2 ± 9.2	**31.7 ± 12.6**	60.2 ± 14.2
	400	22.6 ± 9.2	44.5 ± 12.2	64.9 ± 10.4	14.8 ± 11.5	45.6 ± 18.0	69.6 ± 12.4	22.6 ± 9.2	44.5 ± 12.2	64.9 ± 10.4	13.6 ± 8.6	37.8 ± 14.6	62.1 ± 12.8
	500	25.0 ± 9.0	52.8 ± 12.8	**72.1 ± 7.9**	16.7 ± 12.3	54.9 ± 16.7	**76.5 ± 9.9**	25.0 ± 9.0	52.8 ± 12.8	**72.1 ± 7.9**	15.8 ± 8.8	47.9 ± 14.7	**70.9 ± 9.5**
LR	100	**57.1 ± 12.8**	81.9 ± 3.7	85.1 ± 3.5	**57.8 ± 16.7**	86.0 ± 2.7	87.8 ± 2.9	**57.1 ± 12.8**	81.9 ± 3.7	85.1 ± 3.5	**53.8 ± 15.2**	81.7 ± 3.9	84.8 ± 3.7
	200	61.9 ± 10.2	84.3 ± 2.8	87.7 ± 2.8	63.2 ± 12.9	87.4 ± 2.1	89.5 ± 2.2	61.9 ± 10.2	84.3 ± 2.8	87.7 ± 2.8	59.9 ± 12.1	84.1 ± 2.8	87.6 ± 2.9
	300	63.5 ± 9.9	**84.2 ± 2.0**	88.6 ± 2.1	65.6 ± 12.7	**87.3 ± 1.5**	90.0 ± 1.7	63.5 ± 9.9	**84.2 ± 2.0**	88.6 ± 2.1	62.0 ± 12.0	**83.9 ± 2.1**	88.4 ± 2.2
	400	66.9 ± 9.4	84.8 ± 1.9	89.3 ± 1.8	69.5 ± 10.5	87.8 ± 1.4	90.5 ± 1.6	66.9 ± 9.4	84.8 ± 1.9	89.3 ± 1.8	66.0 ± 11.1	84.6 ± 2.0	89.2 ± 1.9
	500	67.6 ± 7.9	84.5 ± 1.7	**89.3 ± 1.4**	69.2 ± 10.1	87.5 ± 1.2	**90.4 ± 1.3**	67.6 ± 7.9	84.5 ± 1.7	**89.3 ± 1.4**	66.9 ± 9.6	84.3 ± 1.8	**89.2 ± 1.4**
GPC	100	**65.5 ± 5.5**	84.5 ± 3.8	84.5 ± 3.8	**80.7 ± 4.1**	87.5 ± 2.8	87.5 ± 2.8	**65.5 ± 5.5**	84.5 ± 3.8	84.5 ± 3.8	**66.2 ± 6.0**	84.1 ± 4.0	84.1 ± 4.0
	200	65.4 ± 3.5	87.9 ± 2.4	87.9 ± 2.4	80.4 ± 3.1	89.7 ± 1.9	89.7 ± 1.9	65.4 ± 3.5	87.9 ± 2.4	87.9 ± 2.4	66.4 ± 3.8	87.8 ± 2.6	87.8 ± 2.6
	300	61.1 ± 2.7	**89.1 ± 2.0**	89.2 ± 2.0	81.3 ± 2.8	**90.5 ± 1.5**	90.5 ± 1.5	61.1 ± 2.7	**89.1 ± 2.0**	89.2 ± 2.0	62.4 ± 3.3	**89.0 ± 2.1**	89.0 ± 2.1
	400	56.3 ± 2.4	90.1 ± 1.7	90.1 ± 1.7	83.6 ± 1.5	91.3 ± 1.4	91.3 ± 1.4	56.3 ± 2.4	90.1 ± 1.7	90.1 ± 1.7	56.9 ± 3.3	90.0 ± 1.8	90.0 ± 1.8
	500	52.4 ± 1.9	90.7 ± 1.5	**90.7 ± 1.5**	84.1 ± 1.2	91.6 ± 1.2	**91.6 ± 1.2**	52.4 ± 1.9	90.7 ± 1.5	**90.7 ± 1.5**	51.9 ± 2.8	90.6 ± 1.6	**90.6 ± 1.6**

4.2 Comparison Results

Table 1 illustrates the comparison between HDC and other classifiers in edge computing in cybersecurity attack detection. The support vector machine (SVM), multi-layer perceptron (MLP), logistic regression(LR), and Gaussian process classification (GPC) are implemented with the Scikit-Learn library. The classifiers are configured with the default parameters provided by the library. The average and STD of accuracy, precision, recall, and F-score for classifiers are shown for different train data sizes. The result is shown for 1, 5, and 10 iterations.

For the train data sizes of 100, 200, 300, 400, and 500, HDC accuracy is 81.3%, 85.1%, 88.2%, 89.1%, and 89.1% in one-pass learning, 87.4%, 90.9%, 92.6%, 92.8%, and 92.6% in 5 iterations, as well as 91.2%, 92%, 93.2%, 93.7%, and 93.9% in 10 iterations. For each train data size and number of iterations, we compare the best performance of the other classifiers with HDC. For different train data sizes in one-pass learning, the closest to HDC accuracy are 65.5%, 65.4%, 63.5%, 66.9%, and 67.6%, which are 15.8%, 19.7%, 24.7%, 22.2%, and 21.5% below HDC accuracy. According to this result, HDC performs better in one-pass learning than other classifiers, with a difference of more than 15%. In 5 iterations, the closest accuracy to HDC is 84.5%, 87.9%, 89.1%, 90.1%, and 90.7%, which is 2.9%, 3%, 3.5%, 2.7%, and 1.9% below HDC. In 10 iterations, a close match to HDC accuracy appears to be 86.2%, 87.9%, 89.2%, 90.1%, and 90.7%, which is 5%, 4.1%, 4%, 3.6%, and 3.2% below HDC. The results show that HDC is better at classifying even after 5 and 10 iterations compared to other classifiers. When comparing HDC accuracy with other classifiers, the differences between them decreased from one to five iterations but increased again from five to ten iterations. HDC performance is better than other classifiers such as accuracy, with just the precision values being higher than accuracy values when we compare its precision, recall, and F-score to those of other classifiers. GPC and LR have a closer performance to HDC and the performance of HDC and MLP differs significantly. For the HDC, the STDs are decreased and the averages are increased by increasing the number of iterations. Moreover, the averages are improved by increasing the train data size from 100 to 300. However, raising the train data size over 300 only affects averages slightly which means the HDC can be trained by a few samples and doesn't need many samples for the training stage.

Figure 7 demonstrates the results for the steady state situation, which means the classifiers are trained up to the maximum number of iterations and the metrics' average and STD in the last iteration are shown. HDC is trained over 250 iterations and other classifiers that are implemented using the Scikit-Learn library, are trained over a maximum number of iterations that are set by this library. HDC and other classifiers are evaluated over different train data sizes. Therefore, HDC compares with the classifiers which are trained using default parameters that are provided by the Scikit-Learn library. For different train data sizes in the steady state, the HDC accuracy is 93%, 94.8%, 95.6%, 95.7%, and 95.9%, and the STDs are 2.8%, 1.9%, 1.4%, 1.3%, and 1%, respectively. As the number of iterations increases in the HDC model, the standard deviations of the evaluation metrics decrease, and the averages increase. This indicates that the model is becoming more consistent and accurate in its performance as it is trained over more iterations. In addition, increasing the size of the training data from 100 to 300 rises the averages of the evaluation metrics, indicating that the HDC model can benefit from larger amounts of training data up to a certain point. However, after a training data size of 300, increasing the data size further only has a minor impact on the averages of the evaluation metrics. This suggests that the HDC model is able to achieve high accuracy and consistency with relatively small amounts of training data.

For each train data size, we compare the best performance of the other classifiers with HDC. For different train data sizes in the steady state, the closest to HDC accuracy are 88.7%, 90.4%, 92.8%, 92.8%, and 93.7% which are 4.3%, 4.4%, 2.8%, 2.9%,

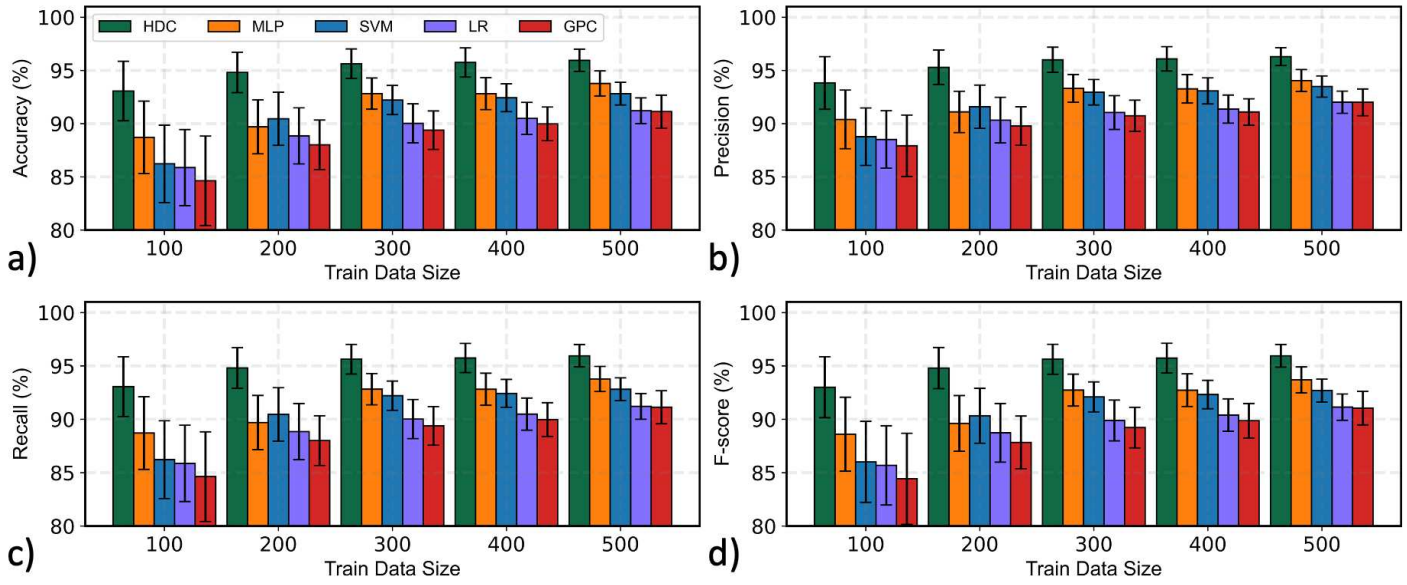

FIGURE 7: THE PERFORMANCE OF HDC FRAMEWORK IN COMPARISON TO MLP, SVM, LR, AND GPC FOR TRAIN DATA SIZES RANGING FROM 100 TO 500 ACCORDING TO THE FOLLOWING EVALUATION METRICS A) ACCURACY, B) PRECISION, C) RECALL, AND D) F-SCORE.

and 2.2% below HDC accuracy. Therefore, HDC does a superior performance than other classifiers when only a few train data are fed to the classifiers. HDC outperforms other classifiers such as accuracy when we compare precision, recall, and F-score with other classifiers, although precision values are higher than accuracy values. Therefore, when only a few train data sizes are provided to the classifiers, HDC outperforms SVM, MLP, LR, and GPC in cybersecurity. In the steady state, MLP is closest to HDC. The performance of GPC and LR is similar to HDC for a small number of iterations; however, in steady state, their performance isn't even close to MLP and SVM, which means they can be trained in a few iterations and with a small amount of train data but cannot compete with HDC.

5. CONCLUSIONS

The Industrial Internet of Things (IIoT) is an emerging technology that has the potential to revolutionize the way industries operate. However, as with any new technology, there are concerns about its security, especially in the face of cyberattacks. This research introduces HDC, a hyperdimensional cognitive learning framework for robust, efficient, and transparent security monitoring. We first represent security-related data into a holographic high-dimensional space to abstract the knowledge. Then, we develop a cognitive learning algorithm capable of security monitoring in a lightweight manner with limited training data. The HDC is integrated with the regeneration capability of hypervectors elements to realize ultra-fast learning of intrusions, making it suitable for detecting new cyberattacks. The experimental results based on the WUSTL-IIOT-2021 dataset show that HDC effectively detects and handles attacks, including backdoor attacks, command injection, denial of service, and reconnaissance. The combination of IIoT and HDC is a promising approach for achieving secure and efficient data processing in industrial settings.

ACKNOWLEDGMENTS

This work was partially supported by Semiconductor Research Corporation (SRC), Office of Naval Research [grant No. N00014-21-1-2225 and N00014-22-1-2067], National Science Foundation (NSF) [grant No. 2127780 and 2312517], Air Force Office of Scientific Research [grant No. FA9550-22-1-0253] and a generous gift from Cisco.

REFERENCES

[1] Boyes, Hugh, Hallaq, Bil, Cunningham, Joe and Watson, Tim. "The industrial internet of things (IIoT): An analysis framework." *Computers in industry* Vol. 101 (2018): pp. 1–12.

[2] Khan, Wazir Zada, Rehman, MH, Zangoti, Hussein Mohammed, Afzal, Muhammad Khalil, Armi, Nasrullah and Salah, Khaled. "Industrial internet of things: Recent advances, enabling technologies and open challenges." *Computers & Electrical Engineering* Vol. 81 (2020): p. 106522.

[3] Jaidka, Himanshu, Sharma, Nikhil and Singh, Rajinder. "Evolution of iot to iiot: Applications & challenges." *Proceedings of the international conference on innovative computing & communications (ICICC)*. 2020.

[4] Breivold, Hongyu Pei and Sandström, Kristian. "Internet of things for industrial automation–challenges and technical solutions." *2015 IEEE International Conference on Data Science and Data Intensive Systems*: pp. 532–539. 2015. IEEE.

[5] Sadeghi, Ahmad-Reza, Wachsmann, Christian and Waidner, Michael. "Security and privacy challenges in industrial internet of things." *Proceedings of the 52nd annual design automation conference*: pp. 1–6. 2015.

[6] Clarke, Gordon, Reynders, Deon and Wright, Edwin. *Practical modern SCADA protocols: DNP3, 60870.5 and related systems*. Newnes (2004).

[7] Hlaing, Zar Chi Su Su and Khaing, Myo. "A detection and prevention technique on sql injection attacks." *2020 IEEE Conference on Computer Applications (ICCA)*: pp. 1–6. 2020. IEEE.

[8] Borgiani, Vladimir, Moratori, Patrick, Kazienko, Juliano F, Tubino, Emilio RR and Quincozes, Silvio E. "Toward a distributed approach for detection and mitigation of denial-of-service attacks within industrial Internet of Things." *IEEE Internet of Things Journal* Vol. 8 No. 6 (2020): pp. 4569–4578.

[9] White, Ruffin, Caiazza, Gianluca, Jiang, Chenxu, Ou, Xinyue, Yang, Zhiyue, Cortesi, Agostino and Christensen, Henrik. "Network reconnaissance and vulnerability excavation of secure DDS systems." *2019 IEEE European symposium on security and privacy workshops (EUROS&PW)*: pp. 57–66. 2019. IEEE.

[10] Hou, Boyu, Gao, Jiqiang, Guo, Xiaojie, Baker, Thar, Zhang, Ying, Wen, Yanlong and Liu, Zheli. "Mitigating the backdoor attack by federated filters for industrial IoT applications." *IEEE Transactions on Industrial Informatics* Vol. 18 No. 5 (2021): pp. 3562–3571.

[11] Wu, Tsu-Yang, Chen, Chien-Ming, Wang, King-Hang and Wu, Jimmy Ming-Tai. "Security analysis and enhancement of a certificateless searchable public key encryption scheme for IIoT environments." *IEEE Access* Vol. 7 (2019): pp. 49232–49239.

[12] Jia, Bin, Zhang, Xiaosong, Liu, Jiewen, Zhang, Yang, Huang, Ke and Liang, Yongquan. "Blockchain-enabled federated learning data protection aggregation scheme with differential privacy and homomorphic encryption in IIoT." *IEEE Transactions on Industrial Informatics* Vol. 18 No. 6 (2021): pp. 4049–4058.

[13] Hodgson, Roderick. "Solving the security challenges of IoT with public key cryptography." *Network Security* Vol. 2019 No. 1 (2019): pp. 17–19.

[14] Vanickis, Romans, Jacob, Paul, Dehghanzadeh, Sohelia and Lee, Brian. "Access control policy enforcement for zero-trust-networking." *2018 29th Irish Signals and Systems Conference (ISSC)*: pp. 1–6. 2018. IEEE.

[15] Li, Shan, Iqbal, Muddesar and Saxena, Neetesh. "Future industry internet of things with zero-trust security." *Information Systems Frontiers* (2022): pp. 1–14.

[16] He, Debiao, Ma, Mimi, Zeadally, Sherali, Kumar, Neeraj and Liang, Kaitai. "Certificateless public key authenticated encryption with keyword search for industrial internet of things." *IEEE Transactions on Industrial Informatics* Vol. 14 No. 8 (2017): pp. 3618–3627.

[17] Chen, Biwen, Wu, Libing, Kumar, Neeraj, Choo, Kim-Kwang Raymond and He, Debiao. "Lightweight searchable public-key encryption with forward privacy over IIoT outsourced data." *IEEE Transactions on Emerging Topics in Computing* Vol. 9 No. 4 (2019): pp. 1753–1764.

[18] Islam, Mohammad Rafsun and Aktheruzzaman, KM. "An analysis of cybersecurity attacks against internet of things and security solutions." *Journal of Computer and Communications* Vol. 8 No. 4 (2020): pp. 11–25.

[19] Hassan, Wan Haslina et al. "Current research on Internet of Things (IoT) security: A survey." *Computer networks* Vol. 148 (2019): pp. 283–294.

[20] Choo, Kim-Kwang Raymond, Gritzalis, Stefanos and Park, Jong Hyuk. "Cryptographic solutions for industrial Internet-of-Things: Research challenges and opportunities." *IEEE Transactions on Industrial Informatics* Vol. 14 No. 8 (2018): pp. 3567–3569.

[21] Huma, Zil E, Latif, Shahid, Ahmad, Jawad, Idrees, Zeba, Ibrar, Anas, Zou, Zhuo, Alqahtani, Fehaid and Baothman, Fatmah. "A hybrid deep random neural network for cyber-attack detection in the industrial internet of things." *IEEE Access* Vol. 9 (2021): pp. 55595–55605.

[22] Latif, Shahid, Zou, Zhuo, Idrees, Zeba and Ahmad, Jawad. "A novel attack detection scheme for the industrial internet of things using a lightweight random neural network." *IEEE Access* Vol. 8 (2020): pp. 89337–89350.

[23] Tran, Minh-Quang, Elsisi, Mahmoud, Mahmoud, Karar, Liu, Meng-Kun, Lehtonen, Matti and Darwish, Mohamed MF. "Experimental setup for online fault diagnosis of induction machines via promising IoT and machine learning: Towards industry 4.0 empowerment." *IEEE access* Vol. 9 (2021): pp. 115429–115441.

[24] Abdullahi, Mujaheed, Baashar, Yahia, Alhussian, Hitham, Alwadain, Ayed, Aziz, Norshakirah, Capretz, Luiz Fernando and Abdulkadir, Said Jadid. "Detecting cybersecurity attacks in internet of things using artificial intelligence methods: A systematic literature review." *Electronics* Vol. 11 No. 2 (2022): p. 198.

[25] Zolanvari, Maede, Teixeira, Marcio A, Gupta, Lav, Khan, Khaled M and Jain, Raj. "Machine learning-based network vulnerability analysis of industrial Internet of Things." *IEEE Internet of Things Journal* Vol. 6 No. 4 (2019): pp. 6822–6834.

[26] Zou, Zhuowen, Alimohamadi, Haleh, Imani, Farhad, Kim, Yeseong and Imani, Mohsen. "Spiking hyperdimensional network: Neuromorphic models integrated with memory-inspired framework." *arXiv preprint arXiv:2110.00214* (2021).

[27] Zou, Zhuowen, Alimohamadi, Haleh, Zakeri, Ali, Imani, Farhad, Kim, Yeseong, Najafi, M Hassan and Imani, Mohsen. "Memory-inspired spiking hyperdimensional network for robust online learning." *Scientific Reports* Vol. 12 No. 1 (2022): p. 7641.

[28] Imani, Mohsen, Zakeri, Ali, Chen, Hanning, Kim, Tae-Hyun, Poduval, Prathyush, Lee, Hyunsei, Kim, Yeseong, Sadredini, Elaheh and Imani, Farhad. "Neural computation for robust and holographic face detection." *Proceedings of the 59th ACM/IEEE Design Automation Conference*: pp. 31–36. 2022.

[29] Zou, Zhuowen, Kim, Yeseong, Imani, Farhad, Alimohamadi, Haleh, Cammarota, Rosario and Imani, Mohsen. "Scalable edge-based hyperdimensional learning system with brain-like neural adaptation." *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*: pp. 1–15. 2021.

[30] Chen, Ruimin, Imani, Mohsen and Imani, Farhad. "Joint active search and neuromorphic computing for efficient data exploitation and monitoring in additive manufacturing." *Journal of manufacturing processes* Vol. 71 (2021): pp. 743–752.

[31] Poduval, Prathyush, Issa, Mariam, Imani, Farhad, Zhuo, Cheng, Yin, Xunzhao, Najafi, Hassan and Imani, Mohsen. "Robust In-Memory Computing with Hyperdimensional Stochastic Representation." *2021 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*: pp. 1–6. 2021. IEEE.

[32] Poduval, Prathyush, Alimohamadi, Haleh, Zakeri, Ali, Imani, Farhad, Najafi, M Hassan, Givargis, Tony and Imani, Mohsen. "Graphd: Graph-based hyperdimensional memorization for brain-like cognitive learning." *Frontiers in Neuroscience* Vol. 16 (2022): p. 5.

[33] Imani, Farhad and Chen, Ruimin. "Latent Representation and Characterization of Scanning Strategy on Laser Powder Bed Fusion Additive Manufacturing." *ASME International Mechanical Engineering Congress and Exposition*, Vol. 86649: p. V02BT02A009. 2022. American Society of Mechanical Engineers.

[34] Chen, Ruimin, Sodhi, Manbir, Imani, Mohsen, Khanzadeh, Mojtaba, Yadollahi, Aref and Imani, Farhad. "Brain-inspired computing for in-process melt pool characterization in additive manufacturing." *CIRP Journal of Manufacturing Science and Technology* Vol. 41 (2023): pp. 380–390.

[35] Wang, Xiaoyi, Eiselmayer, Alexander, Mackay, Wendy E, Hornbaek, Kasper and Wacharamanotham, Chat. "Argus: Interactive a priori power analysis." *IEEE Transactions on Visualization and Computer Graphics* Vol. 27 No. 2 (2020): pp. 432–442.