OnlineHD: Robust, Efficient, and Single-Pass Online Learning Using Hyperdimensional System

Alejandro Hernández-Cano*, Namiko Matsumoto$^{\psi}$, Eric Ping$^{\psi}$, Mohsen Imani†*
*Universidad Nacional Autónoma de México, $^{\psi}$University of California San Diego, †University of California Irvine
*Email: m.imani@uci.edu

***Abstract*—Hyper-Dimensional computing (HDC) is a brain-inspired learning approach for efficient and robust learning on today's embedded devices. HDC supports single-pass learning, where it generates a classification model by one-time looking at each training data point. However, the single-pass model provides weak classification accuracy due to model saturation caused by naively accumulating high-dimensional data. Although the retraining model for hundreds of iterations addresses the model saturation and boosts the accuracy, it comes with significant training costs. In this paper, we propose OnlineHD, an adaptive HDC training framework for accurate, efficient, and robust learning. During single-pass training, OnlineHD identifies common patterns and eliminates model saturation. For each data point, OnlineHD updates the model depending on how similar it is to the existing model, instead of naive data accumulation. We expand the OnlineHD framework to support highly-accurate iterative training. We also exploit the holographic distribution of patterns in high-dimensional space to make OnlineHD ultra-robust against possible noise and hardware failure. Our evaluations on a wide range of classification problems show that OnlineHD adaptive training provides comparable classification accuracy to the retrained model while getting all efficiency benefits that a single-pass training provides. OnlineHD achieves, on average, 3.5× and 6.9× (3.7× and 5.8×) faster and more efficient training as compared to state-of-the-art machine learning (HDC algorithms), while providing similar classification accuracy and 8.5× higher robustness to a hardware error.**

I. Introduction

Internet of Things (IoT) applications collect large amounts of data from various devices and apply machine learning (ML) algorithms to transform that data into actionable knowledge. Running ML algorithms requires significant computational power and storage, resulting in systems that stream most or all the data to the cloud for analysis [1]. This inefficiency comes from the following key technical challenges: (i) Edge devices often do not have sufficient resources to support start-of-the-art ML workloads in real-time [2], [3], [4]. For example, Deep Neural Networks (DNNs) use complex gradient-based computation, which requires memory and resource over the capability of today's embedded devices to enable online learning [5]. (ii) Existing ML algorithms have high sensitivity to noise and failure and often require floating-point precision for training [3]. However, today's devices depend on unreliable battery sources and have major scalability issues that add a large amount of noise to the hardware [6].

To address the efficiency and robustness issues, we exploit Hyper-Dimensional Computing (HDC) as an alternative computational model mimicking "the human brain" in the functionality level [7], [8], [9]. HDC is based on the fact that the brain works with neural activities in high-dimensional space. It maps data points into high-dimensional space and then perform a nearly-linear training to learn a model. HDC is well suited to address learning tasks for IoT systems as: (i) HDC models are computationally efficient and highly parallel at heart to train and amenable to hardware level optimization [10], [11], [12], [13], [14], [15], (ii) HDC models offer an intuitive and human-interpretable model [16], (iii) it offers a complete computational paradigm that can be applied to cognitive as well as learning problems [17], [18], [19], and (iv) it provides strong robustness to noise – a key strength for IoT systems [20], [21].

Existing HDC algorithms are supporting single-pass training, where learning can perform in by single time looking at each training data points [22], [8]. Although this approach enables fast and real-time learning from a stream of data, it provides very weak classification accuracy. For example, for face recognition [23], single-pass training can result in ∼70% classification accuracy, which is 25% lower than state-of-the-art algorithms. To address this issue, prior work introduced the idea of HDC iterative training, called *retraining* [8], [24]. Although retraining boosts HDC classification accuracy to a similar level as the state-of-the-art, it removes advantages that the HDC single-pass model provides. For example, to support retraining, devices require to use large off-chip memory to store all training samples.

We observe that the main limitation of the HDC single-pass training is coming from a naive data accumulating to generate each class hypervector. This causes forgetting and saturation in each class hypervector, where the pattern of common data dominates each class. In this paper, we propose OnlineHD, an adaptive HDC training framework for accurate, efficient, and robust learning. OnlineHD supports single-pass training while ensuring accuracy comparable to the retrained model.

- During single-pass training, OnlineHD identifies common patterns in each class hypervector and eliminates model saturation. For each data point, OnlineHD updates the model depending on how similar is a data point to the existing model. For data points with high similarity, OnlineHD gives very small weights to them during the model update, while dissimilar data points get higher weight depending on how far they are to the current model. OnlineHD adaptive training provides comparable accuracy to the retrained model while getting all benefits that a single-pass model provides.
- OnlineHD framework also supports highly-accurate and efficient iterative training. OnlineHD starts training from a well-developed single-pass model, and in each iteration, it adaptively updates the model based on the distance of each data to the model. OnlineHD adaptive update ensures high classification accuracy as well as fast converge, which is up to 29.6× faster than state-of-the-art retraining approaches [24], [8].
- We study the impact of different data representation on OnlineHD robustness to possible noise in the hardware. We show OnlineHD provides maximum robustness when storing information as holographic distribution of patterns in high-dimensional space, when failures on a dimension will not result in losing the entire data.

We evaluate OnlineHD efficiency on a wide range of

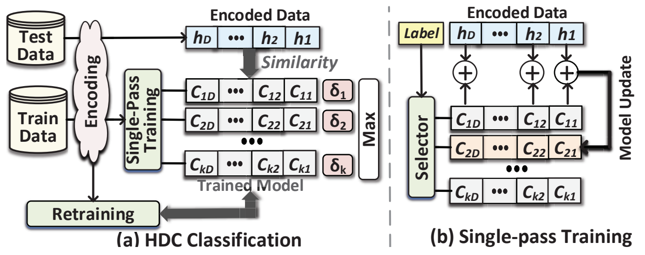


Fig. 1. (a) HDC classification steps, (b) single-pass training.

classification problems. Our evaluation shows that OnlineHD provides, on average, 12.1% higher classification accuracy as compared to the state-of-the-art HDC-based algorithm [22], [24] during single-pass training. OnlineHD also provides better accuracy than the HDC-based algorithms using iterative learning while converging with 13.1× lower number of iterations. In terms of efficiency, OnlineHD provides 3.5× and 6.9× (3.7× and 5.8×) faster and more efficient training than state-of-the-art DNN (HDC) algorithms. In addition, OnlineHD data representation enables 8.5× higher robustness to hardware error as compared to DNN. Our code is available open-source[1].

II. Hyper-Dimensional Classification

Figure 1 shows an overview of Hyper-Dimensional classification (HDC).

Encoding: The first step in HDC is to map each data points into high-dimensional space. The mapping procedure is often referred to as *encoding* (shown in Figure 1). HDC uses different encoding methods depending on data types [22], [25], [24]. The encoded data should satisfy the common-sense principle: data points different from each other in the original space should also be different in the HDC space. For example, if a data point is entirely different from another, the corresponding hypervectors should be orthogonal in the HDC space. Assume an input vector (an image, voice, etc.) in original space $\vec{F} = \{f_1, \ f_2, \cdots, f_n\}$ and $F \in \mathcal{R}^n$. The encoding module maps this vector into high-dimensional vector, $H \in \{0,1\}^D$, where $D >> n$. The following equation shows an encoding method that maps input vector into high-dimensional space [8]: $\vec{\mathcal{H}} = \sum_{k=0}^{\mathcal{D}_{iv}-1} |f_k|_{\in\mathcal{F}} \cdot \vec{\mathcal{B}}_k$, where $\vec{\mathcal{B}}_k$s are randomly chosen hence orthogonal bipolar base hypervectors of dimension $\mathcal{D} \simeq 10k$ to retain the spatial or temporal location of features in an input. That is, $\vec{\mathcal{B}}_k \in \{-1,+1\}^{\mathcal{D}}$ and $\delta(\vec{\mathcal{B}}_{k_1}, \vec{\mathcal{B}}_{k_2}) \simeq 0$, where δ denotes the cosine similarity

Single-pass Training: To find the universal property for training dataset, the trainer module linearly combines hypervectors belonging to each class, i.e., adding the hypervectors to create a single hypervector for each class. Once combining all hypervectors, we treat per-class accumulated hypervectors, called *class hypervectors*, as the learned model. Figure 1b shows HDC functionality during single-pass training. Assuming a problem with k classes, the model represents using: $\mathcal{M} = \{\vec{\mathcal{C}}_1, \vec{\mathcal{C}}_2, \cdots, \vec{\mathcal{C}}_k\}$. For example, after generating all encoding hypervector of inputs belonging to class/label l, the class hypervector $\vec{\mathcal{C}}^l$ can be obtained by bundling (adding) all $\vec{\mathcal{H}}^l$s. Assuming there are $\mathcal{J}$ inputs having label l: $\vec{\mathcal{C}}^l = \sum_j^{\mathcal{J}} \vec{\mathcal{H}}_j^l$

Inference: checks the similarity of each encoded test data with the class hypervector in two steps. The first step encodes the input (the same encoding used for training) to produce a query hypervector $\vec{\mathcal{H}}$. Then, as Figure 1 shows, we compute the similarity (δ) of $\vec{\mathcal{H}}$ and all class hypervectors. Query data gets the label of the class with the highest similarity.

Retraining: HDC classification using single-pass training provides poor classification accuracy. Recently, several work [8], [24] showed the necessity of using iterative training, called retraining, in order to improve HDC classification accuracy (shown in Figure 1). Retraining can boost the HDC model's accuracy by discarding the mispredicted queries from corresponding mispredicted classes and adding them to the right class. The retraining continues for multiple iterations until the classification accuracy (over validation data) has small changes during the last few iterations.

HDC retraining involves large number of iterations to converge. Each iteration is computationally expensive as it requires both associative search and model update. In this paper, our goal is to design a novel HDC training algorithm which can provide classification accuracy of iterative and training efficiency of single-pass training at the same time.

III. OnlineHD Adaptive Learning

In HDC classification, the single-pass model does not well represent the entire dataset. The naive hypervector addition (explained in Section II) results in saturation of class hypervectors by data points with the most common patterns. This saturation hides the information of non-common patterns stored in class hypervectors. For example, consider `cat` and `dog` classification problem. HDC training crates two hypervectors; one representing `cat` and one for `dog`. Lets assume training data consists of 80% of `cat` with similar encoded patterns. This common pattern will dominate the `cat` class and vanishes the pattern of non-common pattern in the class hypervector. Due to model saturation, data points with non-common patterns are likely to miss-classified by the model. This retraining gives higher weight to data with a non-common pattern to have a higher contribution to the final model. In other words, the retraining is equivalent to giving higher weight to non-common inputs in each class hypervector. In this paper, we explore the opportunity of giving weights to each data point during single-pass training.

A. OnlineHD *Single-Pass Training*

We propose OnlineHD, an adaptive training framework for efficient and accurate HDC learning. OnlineHD identifies common patterns during training and eliminates the saturation of the class hypervectors during single-pass training. Instead of naively combining all encoded data, our approach adds each encoded data to class hypervectors depending on how much new information the pattern adds to class hypervectors. If a data point already exists in a class hypervector, OnlineHD will add no or a tiny portion of data to the model to eliminate hypervector saturation. If the prediction matches the expected output, no update will be made to avoid overfitting. Mathematically, OnlineHD adaptive learning is equivalent to the retraining phase, as it provides a higher chance and weight to non-common patterns to represent on the final model. This advantage comes without paying the cost of iterative training.

Figure 2a shows OnlineHD functionality during adaptive initial training. Let's assume $\vec{\mathcal{H}}$ as a new training data point. OnlineHD computes the cosine similarity of $\vec{\mathcal{H}}$ with all class hypervectors. We compute similarity of a data point with $\vec{\mathcal{C}}_i$ as:

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

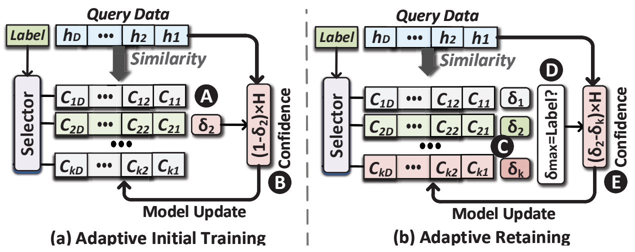


Fig. 2. (a) OnlineHD single-pass training, (b) adaptive iterative training.

$\delta(\vec{\mathcal{H}},\vec{\mathcal{C}^l}) = \frac{\vec{\mathcal{H}}\cdot\vec{\mathcal{C}^l}}{\|\vec{\mathcal{H}}\|\cdot\|\vec{\mathcal{C}^l}\|}$ where $\vec{\mathcal{H}}\cdot\vec{\mathcal{C}^l}$ is a dot product between a query and class hypervector (A). The δ value shows the similarity of a data point to its class hypervector. Instead of naively adding data point to the model, OnlineHD updates the model based on the δ similarity. For example, if an input data has label l, and the most similar class was label l', the model updates as follows (B).

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

where η is a learning rate. A large δ_l indicates that the input is a common data point which is already exist in the model. Therefore, our update adds a very small portion of encoded query to model to eliminate model saturation ($1-\delta_l \simeq 0$). However, small δ_l means that the query has new pattern which does not exist in the model. Thus, the model is updated with a larger factor ($1-\delta_l \simeq 1$). Figure 3 shows OnlineHD classification accuracy during single-pass training. Our evaluation indicates that OnlineHD learns a much more accurate model as compared to the baseline in the first training iteration. OnlineHD advantage comes from its adaptive learning that bolds the impact of all data points in the model, regardless of their frequency and dominance in the model.

B. OnlineHD Iterative Learning

Although single-pass training is suitable for fast and ultra-efficient learning, embedded devices may have enough resources to perform more accurate learning tasks. OnlineHD supports retraining to enhance the quality of the model. Instead of starting to retrain from a naive initial model, OnlineHD retraining starts from the initial adaptive model (explained in Section III-A). OnlineHD initial model already considered the weight of each input data during single-pass training. Therefore, OnlineHD retraining starts from a well-trained initial model with relatively high classification accuracy. This enables OnlineHD to retrain the model with a much lower number of iterations, resulting in fast convergence. Figure 2b shows OnlineHD functionality during adaptive retraining. OnlineHD follows a similar learning procedure as initial training. For each training data point, say H, OnlineHD checks the similarity of data with all class hypervectors in the model (C) and updates the model for each miss-prediction (D). Retraining examines if the model correctly returns the label l for an encoded query $\vec{\mathcal{H}}$. If the model mispredicts it as label l', the model updates as follows (E).

$$\vec{\mathcal{C}_l} \leftarrow \vec{\mathcal{C}_l} + \eta\ (1-\delta_l)\times\vec{\mathcal{H}}$$
$$\vec{\mathcal{C}_{l'}} \leftarrow \vec{\mathcal{C}_{l'}} - \eta\ (1-\delta_{l'})\times\vec{\mathcal{H}} \tag{2}$$

where $\delta_l = \delta(H,\vec{\mathcal{C}_l})$ and $\delta_{l'} = \delta(H,\vec{\mathcal{C}_{l'}})$ are the similarity of data with correct and miss-predicted classes, respectively. This

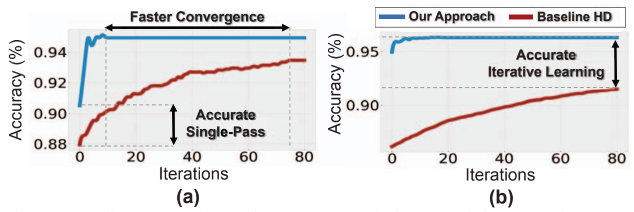


Fig. 3. OnlineHD and baseline accuracy during retraining iterations: (a) speech recognition, (b) activity recognition.

ensures that we update the model based on how far a train data point is miss-classified with the current model. In case of of a very far miss-prediction, $\delta_{l'} \gg 0$, OnlineHD retraining makes a major changes on the mode. While in case of marginal miss-prediction, $\delta_{l'} \simeq 0$, the update makes smaller changes on the model. We also provide separate coefficients for the true and miss-predicted labels, allowing OnlineHD to update each class hypervector independently. Figure 3 shows the classification accuracy of OnlineHD and the baseline HDC during different retraining iterations. Our evaluation indicates that OnlineHD starts learning from higher accuracy using initial adaptive learning. In addition, OnlineHD achieves maximum accuracy with a lower number of iterations as compared to the baseline. Our adaptive retraining also enables HDC to provides higher final accuracy as compared to the baseline HDC algorithm.

IV. OnlineHD Robustness and Efficiency

The technological and fabrication issues in highly scaled technology nodes add a significant amount of noise to both memory and computing units [26], [27]. In addition, embedded devices are often powered based on unreliable battery sources. All these issues result in adding an extra computational error, which degrades the quality of learning. Unfortunately, the existing ML algorithms have very low robustness to noise in hardware. Deep neural networks (DNNs), as the state-of-the-art machine learning algorithms, have very high sensitivity to noise in hardware, especially during training. Earlier work showed that, without enough bit precision, the model training is likely to diverge or provide low accuracy [28] In DNNs, weights represent using fixed-point or floating-point value. An error bit on the exponents or Most Significant Bits (MSBs) results in a major change in the weight value. This makes a traditional floating point or fixed point representation vulnerable and sensitive to an error on the hardware.

A. Hypervector Representation

One of the main advantages of OnlineHD is its high robustness to noise and failure. In OnlineHD, hypervectors are random and holographic with i.i.d. components. Each hypervector stores the information across all its components so that no component is more responsible for storing any piece of information than another. This makes a hypervector robust against errors in its components. OnlineHD efficiency and robustness depend on two parameters: (i) the hypervector dimensionality that determines the hypervector capacity and the level of redundancy, and (ii) the precision of each hypervector element. Increasing dimensionality or precision of elements results in improving the classification accuracy. However, increasing dimensional results in an efficiency issue, while high precision representation reduces the robustness.

In OnlineHD, the encoding module represents original data as a pattern of vectors in high-dimensional space. The goal

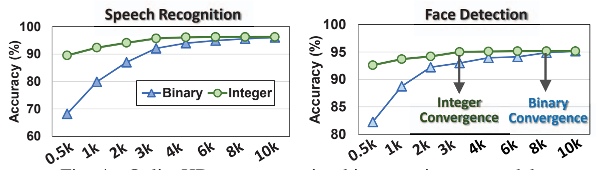


Fig. 4. OnlineHD accuracy using binary or integer models.

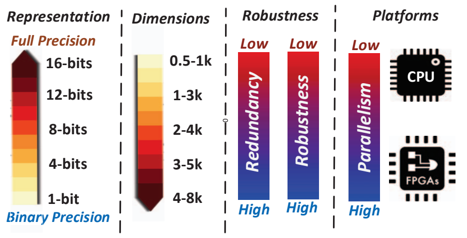
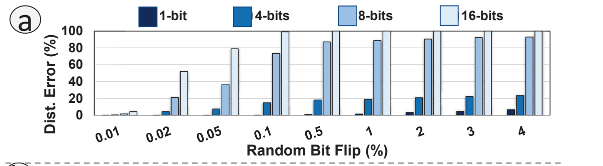


Fig. 5. OnlineHD dimensionality, parallelism, and robustness using different data representations.



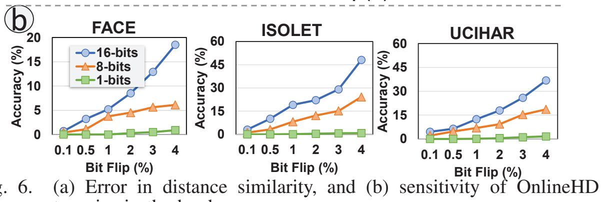


Fig. 6. (a) Error in distance similarity, and (b) sensitivity of OnlineHD accuracy to noise in the hardware.

of this encoding is to preserve the distance of data points in high-dimensional space. The encoding maps original data, $\vec{F} \in R^n$ to D dimensional vectors. The encoded hypervector can be represented using binary ($\vec{\mathcal{H}} \in \{0,1\}^D$) or non-binary values ($\vec{\mathcal{H}} \in R^D$). OnlineHD training and inference preserve the hypervector representation. Using binary hypervector, OnlineHD generates binary class hypervectors and uses Hamming distance as the similarity metric. In contrast, using non-binary representation, OnlineHD training generates a non-binary model and use cosine as a similarity metric. OnlineHD with non-binary hypervector is equivalent to using analog neurons [29], where each dimension gets a high precision value. Figure 4 shows OnlineHD accuracy using binary and integer hypervectors. Our results indicate that OnlineHD with integer elements provides maximum accuracy in much lower dimensionality as compared to using binary hypervector. For example, OnlineHD using 1-bit precision requires $D = 6-8K$ dimensions for maximum accuracy, while OnlineHD using 8-bit precision provides the same accuracy on much lower dimensionality ($D = 2-3K$). We explore impact of hypervector representation on OnlineHD robustness and efficiency.

B. OnlineHD Robustness

As we explained in Section IV-A, OnlineHD has the option of working with relatively low-dimensional vectors with higher precision elements or high-dimensional binary vectors. This representation affects OnlineHD robustness. In binary representation, an error just flips a dimension from 0 to 1 or 1 to 0. This results in tiny changes in the entire hypervector pattern. In contrast, error in high-precision and low-dimensional representation can have a significant impact on OnlineHD robustness. This lower robustness comes from: (i) low dimensionality of OnlineHD that directly translates to less redundancy, thus fewer dimensions can cover up for a possible loss of an element. (ii) Using high precision elements, an error in most significant bits can make major changes in the absolute values. This can cause a corrupted dimension to get a huge value, resulting in a significant change in the hypervector pattern. Figure 5 shows OnlineHD required dimensionality using different bit precision. The results are average dimensions measured over several applications (listed in Section V). Our evaluation shows that increased precision has a direct impact on OnlineHD dimensionality and robustness. Using low precision vectors, OnlineHD requires to use high dimension to provide acceptable accuracy. This translates to providing higher redundancy and higher robustness in OnlineHD.

Figure 6a shows the impact of precision on preserving the distance similarity between the hypervectors in $D = 10,000$ dimensions. The results are reported for vectors with different precision and when the hardware platform has different error rates. The error is modeled by randomly flipping arbitrary bits. Our evaluation shows that vectors with low precision have lower sensitivity to noise, meaning that they preserve the similarity. Using high precision vectors, an error has major changes in the patterns and similarity between the vectors. Figure 4b also shows the impact of hardware noise on OnlineHD classification accuracy. The results show the quality loss as compared to OnlineHD with no error rate. Our evaluation indicates that OnlineHD quality loss increases with hypervector precision. In terms of robustness, it is more desirable to use vectors in high-dimensional space and low-precision values. Ideally, we can use vectors with binary and high dimensional representations that provide maximum redundancy and lowest sensitivity to noise.

C. OnlineHD Efficiency: Hardware Platform

As a light-weight classifier, OnlineHD often runs on embedded devices with limited resources and battery. Embedded CPUs and FPGAs are common devices used in IoT systems [30], [14], [13]. These devices have different resources; thus, they are optimized to run different operations and data representations. As we explained, OnlineHD can work with high-dimensional vectors with low precision elements as well as low-dimensional vectors with high precision elements. In terms of efficiency, selecting between these two configurations depends on the underlying hardware platform. Figure 5 shows OnlineHD dimensionality and parallelism during different bit precision. CPUs are traditionally designed to work with integer values, rather than high-dimensional vectors. In other words, CPUs have limited parallelism, meaning that they can process a few dimensions at a time. This makes CPUs inefficient to process long hypervectors. Therefore, CPUs are more efficient, running OnlineHD in low dimensionality with high precision elements. In contrast, FPGAs consist of several Lookup Tables (LUTs) and Flip-Flops that are significantly efficient for implementing low precision arithmetic operations. For high-precision computation, FPGA computation relies on limited DSP blocks, which are computationally expensive. Therefore, FPGAs are promising to accelerate OnlineHD with high-dimensional and low precision elements. Section V-C explores OnlineHD on different hardware platforms.

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

	n	k	Train Size	Test Size	Description
MNIST	784	10	60,000	10,000	Handwritten Recognition [32]
UCIHAR	561	12	6,213	1,554	Activity Recognition(Mobile) [33]
ISOLET	617	26	6,238	1,559	Voice Recognition [34]
FACE	608	2	522,441	2,494	Face Recognition [23]
PAMAP	75	5	611,142	101,582	Activity Recognition(IMU) [35]
PECAN	312	3	22,290	5,574	Urban Electricity Prediction [36]

V. EVALUATION

A. Experimental Setup

The proposed OnlineHD framework has been implemented with the two co-designed modules, software implementation and hardware acceleration. In software, we verified the effectiveness of the OnlineHD framework on large-scale learning problems. In hardware, We implement OnlineHD training and testing on two embedded platforms: FPGA and CPU. For FPGA, we design the OnlineHD functionality using Verilog and synthesize it using Xilinx Vivado Design Suite [31]. The synthesis code has been tested on the Kintex-7 FPGA. We ensure our efficiency is higher than the automated FPGA implementation at [12]. For CPU, the OnlineHD code has been written in Python and optimized for performance. The code has been implemented on Raspberry Pi (RPi) 3B+ using ARM Cortex A53 CPU. We evaluate OnlineHD accuracy and efficiency on six popular datasets, listed in Table I.

B. OnlineHD Accuracy

State-of-the-art ML Algorithms: We compare OnlineHD classification accuracy with state-of-the-art learning algorithms, including Deep Neural Networks (DNN), Support Vector Machine (SVM), and AdaBoost. The DNN models are trained with Tensorflow, and we exploited the Scikit-learn library to train other ML algorithms. We exploit the common practice of the grid search to identify the best hyper-parameters for each model. Our evaluation shows that OnlineHD provides comparable accuracy to other state-of-the-art algorithms.

State-of-the-art HDC Algorithms: Figure 7 compares OnlineHD classification accuracy with state-of-the-art HDC algorithms [8], [22], [24]. The results are reported when OnlineHD and the baseline are trained using a single-pass and iterative way using $D = 10k$. Our evaluation shows that the existing HDC algorithms provide very low classification accuracy during single-pass training. OnlineHD address this issue by enabling adaptive training, which avoids HDC model saturation. Over all tested applications, OnlineHD single-pass model provides, on average, 12.1% higher classification accuracy compared to the existing HDC algorithms. Interestingly, OnlineHD single-pass accuracy is even 1.2% more accurate than the costly baseline HDC algorithms with iterative training. Since the initial model is already well-trained, the retraining has a lower impact on OnlineHD accuracy. The results show that OnlineHD iterative learning can improve accuracy by 3.0% as compared to a single-pass model.

Partial Training: Figure 8 shows OnlineHD accuracy during single-pass training when we train on a different portion of data. Our evaluation indicates that OnlineHD achieves maximum accuracy using a much lower portion of training data, while the existing HDC algorithms achieve lower accuracy even using the entire dataset. This makes a fast learner and suited for light-weight embedded devices.

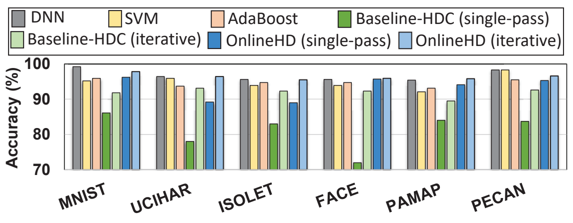


Fig. 7. Comparing OnlineHD accuracy to state-of-the-art.

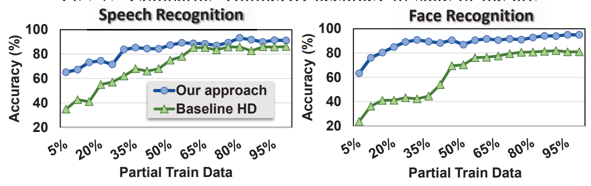


Fig. 8. OnlineHD partial single-pass training.

C. OnlineHD Efficiency

We compare OnlineHD efficiency with DNN and the baseline HDC during training and inference phase on FPGA. All results are reported using binary vectors using dimensionality that both baseline and OnlineHD provide comparable accuracy to DNN. FPGA implementations are optimized to maximize performance by utilizing FPGA resources. During training, OnlineHD provides significantly higher efficiency as compared to DNN and the baseline HDC. This higher efficiency comes from OnlineHD capability in lowering the number of required training iterations. The table in Figure 9a shows the number of training iterations required by OnlineHD and the baseline HDC. OnlineHD adaptive learning significantly reduces the number of required iterations by 13.1×. In contrast, in baseline HDC, the lack of suitable initial model and naive iterative training increases the number of iterations. As compared to DNNs, OnlineHD not only reduces the number of training iterations but also improves the efficiency of a single training iteration. DNNs are using costly gradient operations for training, while OnlineHD computation happens in highly efficient and parallel. As Figure 9a shows, OnlineHD in single-pass further improves the training efficiency by eliminating iterative learning. In this configuration, OnlineHD can train a model with minimum data communications between memory and the computing units. Our evaluation shows that OnlineHD iterative (single-pass) training provides 5.9× and 9.4× (17.1× and 28.7×) faster and higher energy efficiency than DNN.

Figure 9b also compares the OnlineHD inference efficiency with the state-of-the-art. In HDC-based algorithms, i.e., OnlineHD and baseline HD, the inference efficiency directly depends on the hypervector dimensionality. Since OnlineHD and the baseline are using the same dimensions, they provide the same inference efficiency. This efficiency is higher than DNN as its computation relies on limited and costly DSPs on FPGA. In contrast, OnlineHD uses simple bitwise operations, which can be accelerated using FPGA lookup tables (LUTs). Our evaluation shows that OnlineHD provides 3.5× faster and 6.9× higher energy efficiency than DNN.

D. Precision & Platform

We compare OnlineHD efficiency on two embedded platforms: Raspberry Pi 3B+ using ARM CPU and Xilinx Kintex-7 FPGA. Table II lists the number of required OnlineHD dimensions in each bit precision that results in maximum classification accuracy. Table II also reports the average

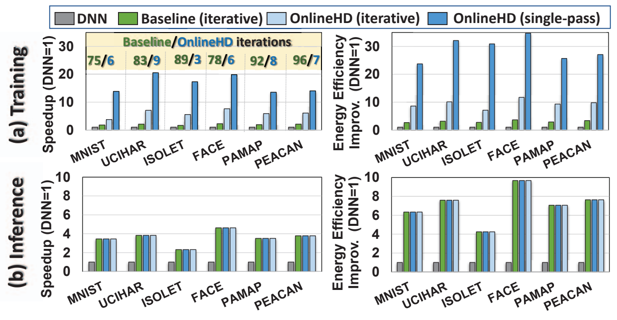


Fig. 9. OnlineHD efficiency vs. state-of-the-art.

TABLE II
IMPACT OF BIT PRECISION ON CPU & FPGA EFFICIENCY

		32-bits	16-bits	8-bits	4-bits	2-bits	1-bits
Dimensions (D)		1.2K	2.1K	3.6K	5.6K	7.5K	8.8K
EDP	**CPU**	1×	1.11×	1.83×	2.24×	3.1×	4.02×
	FPGA	12.90×	11.39×	6.91×	5.64×	4.08×	4.19×

Energy-Delay Product (EDP) of FPGA and CPU running OnlineHD using different hypervector precision. All results are normalized to CPU EDP using hypervectors with 32-bit precision. Our evaluation shows that the CPU provides the highest efficiency using lower-dimensional vectors. This is because CPUs are taking the same number of resources to perform 1-bit or 8-bit arithmetic operations. This limits the amount of parallelism in the CPU. In contrast, FPGAs are more efficient in processing high-dimensional but low precision vectors. The lookup table and flip-flops resources on FPGA can perform several parallel bitwise operations and enable fast and efficient OnlineHD computation. Our goal is to maximum FPGA throughput, where we can process the maximum number of data points at a time. To eliminate off-chip memory to be a bottleneck of computation, FPGA needs to perform maximum computation over each read. We observe that FPGA provides minimum EDP using 2-bit precision. In this precision, OnlineHD maximize FPGA resources while avoiding high precision arithmetic, as the complexity FPGA arithmetic increase exponentially with the bit-width.

E. Robustness to Noise

Table III compares DNN and OnlineHD computation robustness to the noise in the hardware. For OnlineHD, the results are reported using vectors with different precision. The dimensionality of the vector is selected when OnlineHD provides maximum accuracy. Our evaluation shows that OnlineHD provides significantly higher robustness to noise as compared to DNN. In DNNs, the weights are represented as 8-bit values; thus, an error can result in major changes in the weights and classification accuracy. In OnlineHD, information is stored as a holographic distribution of patterns in high-dimensional space. In this representation, all dimensions are equally contributing to storing information. Therefore, failures on data only result in failure on a portion of each hypervector, not losing the entire information. OnlineHD has maximum robustness using vectors with 1-bit precision, while this robustness reduces with the increase in the precision. During 10% hardware error, OnlineHD robustness is 8.5× and 3.8× higher than DNN and OnlineHD using 8-bit precision elements, respectively.

VI. CONCLUSION

In this paper, we propose OnlineHD, an adaptive HDC training framework for accurate, efficient, and robust learning.

TABLE III
ONLINEHD QUALITY LOSS USING NOISY HARDWARE

Hardware Error		*1%*	*2%*	*5%*	*10%*	*15%*
DNN		3.9%	9.4%	16.3%	26.4%	40.0%
OnlineHD	**1-bit**	0.0%	0.0%	0.9%	3.1%	5.2%
	2-bits	0.0%	0.4%	1.4%	4.7%	7.9%
	4-bits	0.3%	1.1%	2.6%	7.3%	11.9%
	8-bits	1.2%	3.7%	5.5%	12.4%	18.7%

During single-pass training, OnlineHD identifies common patterns and eliminates model saturation. We expand OnlineHD to support highly-accurate iterative training. We also exploit the holographic distribution of patterns in high-dimensional space to make OnlineHD ultra-robust against possible noise and hardware failure. Our evaluations show that OnlineHD provides comparable accuracy to the retrained model while providing all efficiency benefits of a single-pass model.

ACKNOWLEDGMENT

This work was partially supported by Semiconductor Research Corporation (SRC) Task No. 2988.001.

REFERENCES

[1] P. Garcia Lopez *et al.*, "Edge-centric computing: Vision and challenges," *SIGCOMM*, vol. 45, no. 5, pp. 37–42, 2015.
[2] M. Brandalero *et al.*, "Multi-target adaptive reconfigurable acceleration for low-power iot processing," *TC*, 2020.
[3] M. Imani *et al.*, "Dual: Acceleration of clustering algorithms using digital-based processing in-memory," in *IEEE/ACM MICRO*, pp. 356–371, IEEE, 2020.
[4] V. Rybalkin *et al.*, "Finn-l: Library extensions and design trade-off analysis for variable precision lstm networks on fpgas," in *FPL*, pp. 89–897, IEEE, 2018.
[5] D. Sapra and A. D. Pimentel, "Constrained evolutionary piecemeal training to design convolutional neural networks," in *IEA/AIE*, 2020.
[6] A. Malek *et al.*, "Odd-ecc: on-demand dram error correcting codes," in *MEMSYS*, pp. 96–111, 2017.
[7] P. Kanerva, "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors," *Cognitive Computation*, vol. 1, no. 2, pp. 139–159, 2009.
[8] M. Imani *et al.*, "A framework for collaborative learning in secure high-dimensional space," in *CLOUD*, pp. 435–446, IEEE, 2019.
[9] M. Imani *et al.*, "Bric: Locality-based encoding for energy-efficient brain-inspired hyperdimensional computing," in *IEEE/ACM DAC*, pp. 1–6, 2019.
[10] M. Imani *et al.*, "Exploring hyperdimensional associative memory," in *HPCA*, pp. 445–456, IEEE, 2017.
[11] B. Khaleghi *et al.*, "tiny-HD: Ultra-Efficient Hyperdimensional Computing Engine for IoT Applications," in *DATE*, IEEE, 2021.
[12] S. Salamat *et al.*, "F5-hd: Fast flexible fpga-based framework for refreshing hyperdimensional computing," in *FPGA*, pp. 53–62, 2019.
[13] B. Khaleghi *et al.*, "Shear er: highly-efficient hyperdimensional computing by software-hardware enabled multifold approximation," in *ISLPED*, pp. 241–246, 2020.
[14] M. Imani *et al.*, "Sparsehd: Algorithm-hardware co-optimization for efficient high-dimensional computing," in *FCCM*, pp. 190–198, IEEE, 2019.
[15] S. Gupta *et al.*, "Thrifty: Training with hyperdimensional computing across flash hierarchy," in *ICCAD*, pp. 1–9, IEEE, 2020.
[16] A. Mitrokhin *et al.*, "Learning sensorimotor control with neuromorphic sensors: Toward hyperdimensional active perception," *Science Robotics*, vol. 4, no. 30, 2019.
[17] M. Imani *et al.*, "Revisiting hyperdimensional learning for fpga and low-power architectures," in *HPCA*, IEEE, 2021.
[18] M. Nazemi *et al.*, "Synergiclearning: Neural network-based feature extraction for highly-accurate hyperdimensional learning," *arXiv preprint arXiv:2007.15222*, 2020.
[19] F. Yang *et al.*, "Adversarial attacks on brain-inspired hyperdimensional computing-based classifiers," *arXiv preprint arXiv:2006.05594*, 2020.
[20] Y. Kim *et al.*, "Geniehd: Efficient dna pattern matching accelerator using hyperdimensional computing," in *DATE*, IEEE, 2020.
[21] S. Salamat *et al.*, "Accelerating hyperdimensional computing on fpgas by exploiting computational reuse," *IEEE TC*, 2020.
[22] A. Rahimi *et al.*, "A robust and energy-efficient classifier using brain-inspired hyperdimensional computing," in *ISLPED*, pp. 64–69, ACM, 2016.
[23] A. Angelova *et al.*, "Pruning training sets for learning of object categories," in *CVPR*, IEEE, 2005.
[24] M. Imani *et al.*, "A binary learning framework for hyperdimensional computing," in *DATE*, pp. 126–131, IEEE, 2019.
[25] A. Rahimi *et al.*, "Hyperdimensional biosignal processing: A case study for emg-based hand gesture recognition," in *ICRC*, pp. 1–8, IEEE, 2016.
[26] A. Rahmani *et al.*, "Reliability-aware runtime power management for many-core systems in the dark silicon era," *TVLSI*, vol. 25, no. 2, pp. 427–440, 2016.
[27] C. Li *et al.*, "A scalable design of multi-bit ferroelectric content addressable memory for data-centric computing," in *IEDM*, IEEE, 2020.
[28] S. Han *et al.*, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.
[29] G. Indiveri *et al.*, "Neuromorphic architectures for spiking deep neural networks," in *IEDM*, pp. 4–2, IEEE, 2015.
[30] M. Blott *et al.*, "Finn-r: An end-to-end deep-learning framework for fast exploration of quantized neural networks," *TRETS*, vol. 11, no. 3, 2018.
[31] T. Feist, "Vivado design suite," *White Paper*, vol. 5, 2012.
[32] D. Ciregan *et al.*, "Multi-column deep neural networks for image classification," in *CVPR*, pp. 3642–3649, IEEE, 2012.
[33] D. Anguita *et al.*, "Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine," in *AAL*, pp. 216–223, Springer, 2012.
[34] "Uci machine learning repository." http://archive.ics.uci.edu/ml/datasets/ISOLET.
[35] A. Reiss and D. Stricker, "Introducing a new benchmarked dataset for activity monitoring," in *ISWC*, pp. 108–109, IEEE, 2012.
[36] "Pecan street dataport." https://dataport.cloud/.