SemiHD: Semi-Supervised Learning Using Hyperdimensional Computing

Mohsen Imani†, Samuel Bosch‡, Mojan Javaheripi*, Bita Rouhani*, Xinyu Wu†,
Farinaz Koushanfar*, and Tajana Rosing†*
†CSE Department, *ECE Department, University of California San Diego, La Jolla, CA, USA
‡ Integrated Systems Laboratory, EPFL, Lausanne, Switzerland
{moimani, mojavahe, bita, x6wu, fkoushanfar, tajana}@ucsd.edu; samuel.bosch@epfl.ch

***Abstract*—In the Internet of Things (IoT), the large volume of data generated by sensors poses significant computational challenges in resource-constrained environments. Most existing machine learning algorithms are unable to train a proper model using a significantly small amount of labeled data available in practice. In this paper, we propose SemiHD, a novel semi-supervised algorithm based on brain-inspired HyperDimensional (HD) computing. SemiHD performs the cognitive task by emulating *neuron's activity* in high-dimensional space. SemiHD maps data points into high-dimensional space and trains a model based on the available labeled data. To improve the quality of the model, SemiHD iteratively expands the training data by labeling data points which can be classified by the current model with high confidence. We also proposed a framework which enables users to trade accuracy for efficiency and select the desired reliability of the model in detecting out of scope data. We have evaluated SemiHD's accuracy and efficiency on a wide range of classification applications and two types of embedded devices: Raspberry Pi 3 and Kintex-7 FPGA. Our evaluation shows that SemiHD can improve the classification accuracy of supervised HD by 10.2% on average (up to 27.3%). In addition, we observe that SemiHD FPGA implementation achieves 7.11× faster and 12.6× energy efficiency as compared to the CPU implementation.**

***Index Terms*—Brain-inspired computing, Semi-supervised learning, Machine learning, Energy Efficiency**

I. Introduction

Data is growing faster than ever before and by the year 2020, about 1.7 megabytes of new information will be created every second for every human being on the planet [1]. With the emergence of the Internet of Things (IoT), many applications run machine learning (ML) algorithms to perform cognitive tasks. In the current state-of-the-art systems, the embedded devices transmit all the raw data to the cloud for processing. However, the total number of devices continually sensing and sending information skyrockets, causing severe network congestion, waste of energy and violation of real-time response constraints [2]–[4]. Furthermore, transmitting raw data off devices neglects privacy which is a key concern in IoT applications that often operate in homes or other sensitive areas [5].

Due to the above-mentioned limitations, it is highly desired to devise machine learning algorithms that can run locally on resource-constrained embedded devices. However, handling the large volume of data generated by sensors poses (at least) two sets of challenges: (i) The existing learning algorithms, e.g., deep neural network, leveraged in smart applications usually require significant computational resources and memory to conform to application-specific requirements. As such, the pertinent hardware constraints hinder the usability of machine learning in a wide variety of real-life applications where device resources and energy are limited [6], [7]. Clearly, there is a pressing need for alternative learning paradigms which can run directly on constrained devices without significantly sacrificing the predictive capacity of state-of-the-art machine learning techniques. (ii) While data is generated at an unprecedented rate, the majority of the sensor data is unlabeled. As a result, conventional supervised learning algorithms which require a large volume of labeled data to train, fail to obtain desired prediction accuracy in such scenarios. Therefore, development and realization of customized semi-supervised learning algorithms on embedded hardware is necessary to fully exploit the unlabeled data.

HD computing emerged from theoretical neuroscience as a computationally tractable, but mathematically rigorous way to model human cognition [8]. At the high level, HD computing represents objects as points known as hypervectors, in a very high dimensional space known as a hyperspace. HD exploits mathematical properties of random numbers and high dimensional spaces to develop a model of computation which operates on these hypervectors. We argue that HD computing is well suited to address learning tasks for IoT networks for three key reasons [9]. First, HD models are computationally efficient to train and amenable to hardware level optimization [10], [11]. Second, HD is a fast learning algorithm which can perform the classification with much less labeled data [12], [13]. Third, HD models offer an intuitive and more human-interpretable alternative to deep learning based techniques. Finally, the mathematics of HD computing provide strong robustness to noise. This is a critical feature in sensor networks where wireless communication is often unreliable and data loss or corruption is possible. However, HD computing cannot provide high classification accuracy when the size of the labeled data is small.

In this paper, we propose SemiHD, a novel semi-supervised HD computing algorithm which enables self-training in high-dimensional space using very small amounts of labeled data. SemiHD performs the classification task by mapping data points into high-dimensional space and generating a model based on the available labeled data. Here, we listed the main contribution of the paper:

- To the best of our knowledge, SemiHD is the first semi-

978-1-7281-2350-9/19/$31.00 ©2019 IEEE

supervised learning algorithm based on the brain-inspired HD computing algorithm. SemiHD starts the training with the labeled data and iteratively expands the training data by labeling data points which can be classified with high confidence. This gradually improves the quality of the model by enabling learning task on a large fraction of the training data.
- We design a novel framework which enables users to trade accuracy and efficiency before running the application just based on the application parameters. Our framework also enables users to choose the desired reliability of the model at runtime, in order to identify data points which are out of the classification scope.
- We devise an efficient FPGA implementation to accelerate SemiHD computation. Our implementation supports our binary and non-binary models using a fully pipelined architecture.
- We examine the accuracy/efficiency of SemiHD on a wide range of classification applications. Our evaluation shows that SemiHD can improve the classification accuracy of supervised HD by 10.2% on average (up to 27.3%). In addition, we observe that SemiHD FPGA implementation achieves 7.11× faster and 12.6× energy efficiency as compared to the best CPU implementation.

II. Background and Motivation

A. Semi-supervised

In many real-life applications, the portion of labeled data is quite smaller compared to the high volume of unlabeled samples. This is partially due to the high labor associated with hand-labeling. Furthermore, with the unprecedented rate of data generation in smart IoT devices and the pertinent real-time requirement, it is often not possible to label the data on-the-go. In supervised learning, only labeled data are useful whereas unsupervised learning merely relies on learning from unlabeled data. Rather than relying solely on one data type, *semi-supervised* learning simultaneously benefits from advantages of the two approaches by leveraging both labeled and unlabeled data to enhance learning.

Several approaches for semi-supervised learning have been proposed in literature, namely, self-training [14], generative models [15], S3VMs [16], graph-based algorithms [17], and multi-view methods [18]. The above-mentioned algorithms have their advantages and drawbacks. For example, generative models can provide an effective probabilistic framework that can enjoy a high accuracy if the underlying model is close to the true distribution of the data. However, the computational complexity associated with this group of semi-supervised algorithms often hinders efficient deployment on embedded devices. In this paper, we utilize self-training as an effective and light-weight semi-supervised learning approach to ensure compatibility with extreme resource constraints. It is known that self-training can lead to weak classifiers since mistakes in the earlier stages of the training algorithm can reinforce themselves. To overcome this challenge, we incorporate a confidence check into our framework to prevent mislabeled samples from interfering in the training process.

B. Hyperdimensional Computing

Hyperdimensional computing is motivated by the observation that the human brain operates on high dimensional representations of data [8]. This high-dimensional space is referred to as a hyperspace, while points in the space are known as hypervectors (e.g., $D = 10{,}000$). The elements of a hypervector are typically either bits (i.e. 0,1) or real numbers. Because of their high-dimensionality, any randomly chosen pair of hypervectors will be nearly orthogonal [19]. In most applications, data input is in the form of low-dimensional signals. The process of transforming sensor input into a high-dimensional representation suitable for HD is known as encoding. In some cases, encoding can be done simply by randomly generating a hypervector. For example, to encode a text document, work in [20]–[22] generates a random hypervector for each letter of the alphabet and then represents words as sequences of letters. Such an encoding scheme is suitable when the relationship between input data is nominal (e.g. there is no notion of distance) and takes a finite number of values. Prior work has also proposed HD encoding methods for different applications and data types, including speech recognition [19], DNA sequencing [23], activity recognition [24], and clustering [25]. For example, work in [13], [26] introduced the idea of fully binary learning in HD computing. There are also several research that try to accelerate HD computing in both existing [11], [27], [28] and emerging hardware [22], [29].

Regardless of the data type and the encoding, HD computing cannot train a proper model with a small size of training data. Our primary evaluation on 18 popular datasets listed in Section VI-B shows that HD computing accuracy highly depends on the amount of labeled data. For example, training HD computing with 10% of labeled data reduces the HD classification accuracy by 34.6% as compared to accessing the entire training data. In practice, there are very few amounts of labeled data, thus the algorithm should be able to get the maximum accuracy using such a small portion of data. In this paper, we propose a novel self-learning algorithm for HD computing which enables training with significantly lower data sizes.

III. SemiHD: HD Self-Training

A. Overview

In this paper, we propose SemiHD, a self-training approach which enables Hyperdimensional (HD) computing to learn to work with an extremely low number of labeled data points. Figure 1 shows an overview of the proposed SemiHD algorithm. Our approach trains HD in the following steps:

- **Encoding:** The first step is to encode all data points into high-dimensional space, where each data represents a vector with $D = 10{,}000$ dimensions (Ⓐ). This encoding applies on both labeled and unlabeled data points and can be different depending on the data type (details explained in Section III-B)
- **HD Training:** SemiHD starts training a model based on the available encoded labeled data. This training can be performed by the accumulation of all encoded data points corresponding to a class. The result of training is

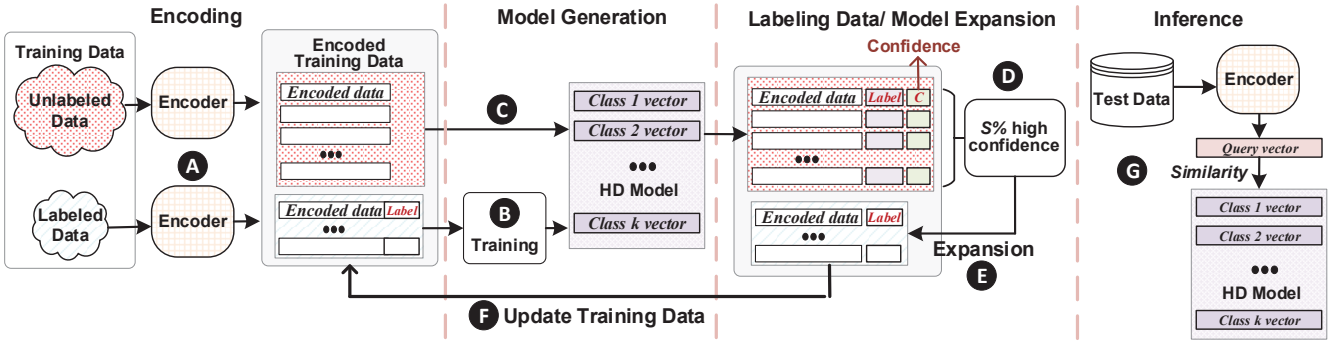

Fig. 1. Overview of SemiHD framework supporting self-training in high-dimensional space.

k hypervectors, each representing one of the classes (Ⓑ). The details of the training are explained in Section III-C.

- **Predict Label of Unlabeled Data:** The trained HD model is ready to be used at the inference/test phase. However, the goal of SemiHD is to improve the classification accuracy by expanding the labeled data. SemiHD exploits the trained HD model in order to assign a label to each unlabeled data. Labeling is similar to the inference task that we check the similarity of each unlabeled data with all the class hypervectors. Each data point gets a label of a class which it has the highest similarity with it (Ⓒ).
- **Determine Prediction Confidence:** We use the difference between the highest top two matches across all class hypervectors to denote the confidence level of the prediction. Top two similarities close to each other indicates that SemiHD has less confidence in its prediction (Ⓓ).
- **Expand Labeled Data:** SemiHD selects and adds $S\%$ of unlabeled data with the highest confidence to labeled data. The value of S is an expansion rate which determines the amount of changes SemiHD makes on the labeled data (Ⓔ).
- **Convergence Condition:** We start training a new HD model based on the expanded training data by going to again to "HD training step" (Ⓕ). SemiHD also checks for the convergence by looking at the classification accuracy during the last three iterations. SemiHD stops the iterative process if the accuracy does not change more than 0.1%.
- **Inference:** After convergence, the model can be used to perform the inference task. In HD computing, the inference can be performed by checking the similarity of each encoded test data with the trained model (Ⓖ).

B. Encoding

Figure 2a shows the overview of HD computing performing the classification task on high-dimensional space. HD consists of three main modules: encoding, training, and associative search. SemiHD functionality is independent of the encoding module. In this work, we use the most general encoding approach which maps vectors $F = \{f_1, f_2, \ldots, f_n\}$, with n features ($f_i \in \mathbb{N}$), into $H = \{h_1, h_2, \ldots, h_D\}$ with D dimensions ($h_i \in \{0,1\}^D$) [8], [30]. Figure 2b shows the functionality of the HD encoding module. This encoding finds the minimum

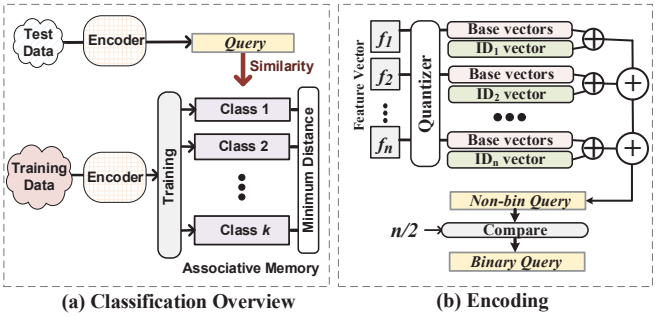

Fig. 2. (a) HD classification Overview, (b) encoding module.

and maximum feature values and quantizes that range into m levels. Then, it assigns a random binary hypervector with D dimensions to each of the quantized levels $\{L_1, \ldots, L_m\}$. Similarly, the encoding module assigns a random binary hypervector to each feature index, $\{ID_1, \ldots, ID_n\}$, where $ID \in \{0,1\}^D$. The encoding happens by combining the feature values over different indices, where the hypervectors corresponding to the feature indices preserves the position of features in a combined set:

$$H = ID_1 \oplus \overline{L}_1 + ID_2 \oplus \overline{L}_2 + \ldots + ID_n \oplus \overline{L}_n.$$

where H is the encoded hypervector and $\overline{L}_i$ is the hypervector corresponding to the i-th feature of vector F. The encoded hypervector can be binarized by applying majority function on each of the dimensions. All elements with a larger value than $n/2$ will be assigned to "1" while other dimensions get "0" value.

C. HD Training

In HD, training is performed in high-dimensional space by element-wise addition of all encoded hypervectors in a class. For example, in a face detection application, the training module accumulates all hypervectors corresponding to "face" and "no-face" in two different hypervectors. In general, the result of training will be k hypervectors with D dimensions, where k is the number of classes. For example, the i^{th} class hypervector can be computed as: $\mathbf{C^i} = \sum_{\forall j \in class_i} \mathbf{H_j}$

D. Data Labeling

Given the limited size of the labeled data, the original predictions made can be inaccurate. Early mistakes made on unlabeled data predictions can reinforce themselves as there is no distinguishing between the originally given labels and the labels generated by the classifier itself. Therefore, we introduce a confidence level on each prediction on unlabeled data. In SemiHD, we apply self-training as a wrapper method combined with hyperdimensional computing. Algorithm 1 describes the steps of self-training with confidence. Let V be the hypervector representing one unlabeled data point, i be the predicted label and j be the second most likely guess. Let C_i denotes the i^{th} class hypervector and C_j denotes the j^{th} class hypervector. Here we define the confidence level of our prediction on the data point as the difference between the similarities of V with C_i and C_j classes:

$$\Delta\delta_{(V)} = \delta(V,C_i) - \delta(V,C_j)$$

where δ is *cosine* or *Hamming distance* similarity for non-binary and binary models respectively. In Section VI-F, we explain that the selection of the model type depends on the underlying hardware (CPU vs. FPGA).

Only the predictions that have the highest confidence are considered for the labeled data expansion. SemiHD adds $S\%$ of unlabeled data with the highest confidence level into the labeled data. After that, SemiHD starts training a new HD model based on the expanded training data. The process of data expansion continues iteratively until the classification accuracy changes less than $\varepsilon = 0.1\%$ during three consecutive iterations. Figure 3 shows the impact of retraining on the classification accuracy of three different datasets ($S = 5\%$), when SemiHD only access to 10% labeled data. Our evaluation shows that SemiHD significantly increases the HD classification accuracy during different retraining iterations. This accuracy improvement saturates after 10-15 iterations, thus these applications can have early convergence, e.g., 10 iterations for *bupa* dataset which aims to detect patients liver disorder.

E. Inference

SemiHD uses encoding and associative search for classification. First, SemiHD uses the same encoding module, explained in Section III-B, to map a test data point to a *query hypervector*. In HD space, the classification task then is performed by checking the similarity of the query with all class hypervectors. Each data point is assigned to a class which it has the highest similarity. When HD information is stored as the pattern of non-binary values, the cosine is a suitable metric for similarity check. For the HD model with binary values, SemiHD uses Hamming distance as the similarity metric.

IV. SemiHD Framework Support

We design a novel framework which enables users to trade SemiHD accuracy and efficiency. Figure 4 shows an overview of the proposed framework. In this framework, users can select between high accuracy or high-performance computation and also they can specify the level of reliability that they expect the model to provide. Here we define the reliability as a capability of SemiHD in detecting uncorrelated test data which are out

Algorithm 1 SemiHD algorithm with confidence

```
while unlabeledData.size > 0 do
    Train ({C_1, ..., C_k}, labeledData) V ∈ unlabeledData
    Label ← argmax_{j=1:k}{δ⟨V, C_i⟩}
    Δδ_V ← argmax_{j=1:k}{δ⟨V, C_i⟩} − argmax_{j≠i}{δ⟨V, C_i⟩}
    V ∈ unlabeledData
    Find(Δδ_V, S)
    labeledData ← (V, Label)
    Remove ← (V, unlabeledData)

end while
```

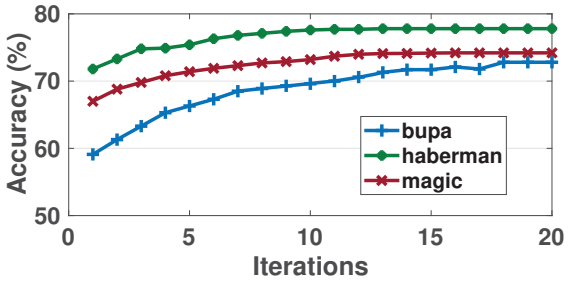

Fig. 3. Impact of retraining on the SemiHD accuracy.

of the classification scope. Our framework gets applications and user specifications as inputs. Application's inputs are the number of features (n), the number of classes (k), and the number of labeled/unlabeled data points. Similarly, users can select between the high efficient or high accurate classification as well the level of reliability that they expect from SemiHD model. In the following, we explain how *Design Analyzer* selects the SemiHD training/inference parameters in order to satisfy the user's expectation.

A. Accuracy vs Efficiency

SemiHD classification accuracy depends on two parameters: dimensionality of the hypervectors and confidence threshold during training.

Dimensionality: This parameter affects both training and inference efficiency since SemiHD requires longer sized vectors. Large dimensionality increases the classification accuracy, but this accuracy saturates when vector size is larger than required. Our framework identifies a mathematical approach to identify the required dimensionality of the hypervectors for each application before training. Theoretically, each hypervector can store a limited amount of information. Considering a hypervector with D dimensions in an application with k classes, each class hypervector can store maximum information of $D/2k$. In other words, each class hypervector can be stored at most $D/2k$ orthogonal hypervectors. To store more information in a class hypervector, we require a class hypervector with higher dimensionality. Assuming all encoded training data is orthogonal (worst-case scenario), we can estimate the dimensionality as:

$$argmax_{i=1:k}\{\#\ labeled\ data_i\} < D/2K$$

The maximum number of labeled data in a class determines the dimensionality of hypervectors. SemiHD exploits this equation to estimate the dimensionality of the encoded data assuming

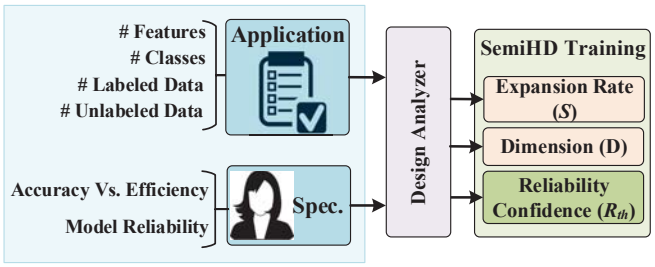

Fig. 4. Framework supporting accuracy-efficiency trade-off and model reliability.

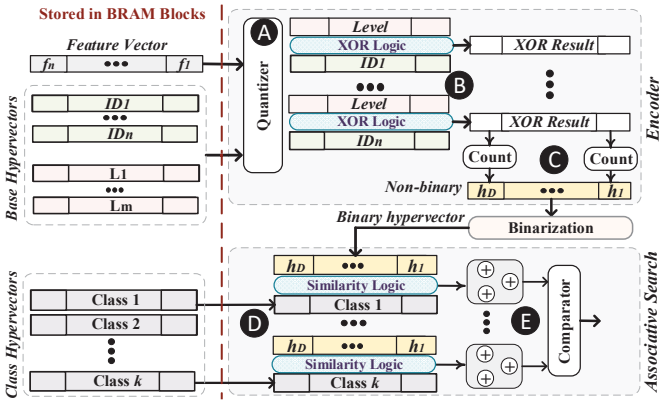

Fig. 5. FPGA implementation of SemiHD inference.

all data points are orthogonal. In practice, training data points are not dissimilar, thus SemiHD estimates the required dimensionality based on the distribution of the training data points. Datasets with a wide distribution ($\sigma \gg$) require dimensionality close to a maximum bound, while datasets with a narrow distribution require much lower dimensionality.

Expansion Rate: In SemiHD, the expansion rate, S, provides a trade-off between the training execution time and classification accuracy. A larger expansion rate requires fewer iterations to converge. In every iteration, SemiHD adds a larger fraction of unlabeled data into the training data, resulting in more changes in the model. In contrast, a small expansion rate leads to minor changes in the model, thus providing a slow training process. Expansion rate also affects SemiHD classification accuracy. A large expansion rate increases the amount of fluctuation in the model during retraining which increases the possibility of divergence. In contrast, a small expansion rate results in higher accuracy but with the slow training process.

B. Model Reliability

One major advantage of HD computing is having an interpretative model. During inference, SemiHD selects a class with the highest similarity to encoded test data as an output. However, in practice, none of the classes might correspond to input data. Current HD computing algorithms are unable to provide a correct response to these queries. For example, in voice recognition problem, an input data with "cat" image will match with one of the class hypervectors, while in reality, it does not correspond to any of the classes. We address this issue by proposing an approach which decides if an input data is irrelevant to the classes. During inference, SemiHD finds a class hypervector which has the highest similarity to an encoded test data. If the similarity of a data point is less than a reliability threshold value, called R_{th}, SemiHD considers such input data to be out of the classification scope. In conventional HD computing approaches, the model is not reliable ($R_{th} = 0$) meaning that a data point will always match with a class with the highest similarity. However, we can select larger confidence, e.g., $R_{th} = 0.8$, to ensure that if a data point is less than 0.8 similar to class hypervector, it is considered as out of scope data. SemiHD does not achieve model reliability freely. Increasing the reliability of the model can result in losing classification accuracy, as several relevant data points may be incorrectly detected as out of scope data. In Section VI-D, we explore the impact of reliability confidence on SemiHD.

V. SemiHD Acceleration

A large portion of SemiHD inference devotes to the encoding module mapping query data points into high-dimensional space. Since the existing processing cores, e.g, CPUs, are not designed to work with long binary vectors, here we design an FPGA implementation of SemiHD. FPGA can be programmed to support SemiHD operations with high efficiency using available lockup table and DSP resources. In the following, we explain the details of SemiHD implementation.

Encoding: Although binary and non-binary models use different associative searches, the encoding is common among both models. Figure 5 shows an overview of the FPGA implementation of the encoding module. The first step is to quantize the feature vector, F, and assign it to one of the m level hypervectors (Ⓐ). Next, our design applies an XOR operation between each ID and level hypervector (Ⓑ). The result of XOR operations are then accumulated using counter blocks (Ⓒ). The quantization, XOR, and count operations can all be implemented using lookup tables (LUTs) resources of FPGA. The Performance of the encoding depends on the number of features and the dimensionality of the encoded data, which determine the required size of the XOR array and the number/size of counter blocks. For the binary model, SemiHD maps each dimension of the encoded data into binary using comparator blocks which is also implemented by LUTs.

Binarized Associative Search: The associative search on the binary model can be performed using another XOR array which computes the similarity of each class hypervector with an input query (Ⓓ). The result of XOR determines the number of bit differences between each query and class hypervectors which are computed using a set of tree-based adders (Ⓔ). Finally, a comparator block finds a class which has the maximum similarity with the query hypervector. We implemented SemiHD codes in order to maximize resource utilization and throughput. Since both encoding and associative search blocks are sharing the same resources (LUTs), our implementation partially generates a part of encoded data which can be used for the associative search. Depending on the application, i.e.,

the number of features and number of classes, our implementation balances the number of resources such that the maximum number of dimensions can be computed at each cycle. This is an important fact since assigning a large portion of the LUTs to the encoding block results in generating a huge amount of dimensions which cannot be used by the associative search block. Similarly, assigning a large portion of FPGA resources to associative search blocks results in lower resources for encoding which reduces SemiHD throughput.

Non-Binary Associative Search: For non-binary models, SemiHD compares the similarity of the query and class hypervectors using dot product. This similarity search can be performed using DSPs available on the FPGA block. Since the encoding and associative search blocks do not share any resources, the throughput of the encoding module depends on the dimensions that can be generated and consumed by the encoding and associative search. The number of dimensions depends on the available LUTs and DSPs. Our evaluation shows that in applications with large feature size, the encoding is the main bottleneck of the computation. In contrast, for applications with a large number of DSPs, the associative search limits the maximum throughput.

VI. Experimental Results

A. Experimental Setup

We have implemented SemiHD framework including encoding, training, and inference in high-dimensional space using C++. We evaluated the system on two embedded platforms: Raspberry Pi 3 with ARM Cortex A53 CPU and Kintex-7 FPGA. For CPU, the SemiHD code has been written in C++ and optimized for performance and the power is measured using Hioki 3334 power meter. For FPGA, we fully implement SemiHD inference using Verilog. We verify the timing and the functionality of the sparse models by synthesizing them using Xilinx Vivado Design Suite [31]. The synthesis code has been implemented on the Kintex-7 FPGA KC705 Evaluation Kit. We compare the SemiHD accuracy with other light-weight classifiers including pruned Decision Tree, Naive Bayes, and Support Vector Machine. We exploited Scikit-learn library [32] for the model training and testing and used grid search to find the best hyperparameters.

B. SemiHD Accuracy vs Other Classifiers

We compare the accuracy of SemiHD with the other light-weight classifiers used for semi-supervised learning. To achieve maximum accuracy, all algorithms have been trained for 40 iterations. We report SemiHD accuracy for two cases: using non-binarized and binarized models. The SemiHD model binarization happens once after the training. Using non-binarized model, SemiHD needs to use cosine similarity to find the most similar class at the inference. Model binarization simplifies the SemiHD similarity to a hardware-friendly metric such as Hamming distance. This model binarization comes at the expense of losing the classification accuracy. For SemiHD, we used $D = 4000$ and expansion rate of $S = 5\%$. All results are reported for 18 popular datasets [33] listed in Table I. Our evaluation shows that SemiHD using non-binarized model can provide significantly higher classification accuracy as compared to other approaches. For example, SemiHD can achieve 4.8% and 7.2% higher accuracy as compared to an SVM and DT respectively. In addition, we observe that SemiHD using binarized model can still provide comparable accuracy to SVM, while its accuracy is 2.3% and 8.8% higher than DT and NB respectively.

TABLE I
Classification accuracy of SemiHD and other algorithms self-training on 10% labeled data.

Datasets	# Ex.	DT	NB	SVM	SemiHD Non-bin	SemiHD Binary
abalone	4174	19.4%	22.4%	21.7%	42.4%	37.8%
chess	3196	85.1%	58.9%	89.7%	97.7%	94.3%
coil2000	9822	55.7%	53.1%	60.9%	94.1%	88.5%
contraceptive	1473	95.6%	80.3%	89.8%	88.0%	82.1%
magic	19020	82.1%	72.3%	83.8%	79.5%	74.1%
marketing	8993	27.7%	26.2%	23.4%	33.6%	30.2%
mushroom	8124	99.6%	92.4%	99.4%	98.4%	91.9%
page-blocks	5472	95.2%	86.2%	94.0%	96.8%	90.8%
penbased	10992	89.0%	81.5%	97.5%	97.9%	92.8%
phoneme	5404	78.0%	73.5%	82.5%	78.3%	73.8%
satimage	6435	80.3%	78.4%	82.4%	78.5%	74.0%
segment	2310	89.9%	69.2%	91.2%	89.8%	85.0%
spambase	4597	86.6%	83.6%	85.4%	90.5%	85.1%
texture	5500	82.6%	72.6%	96.4%	99.2%	94.1%
thyroid	7200	99.1%	90.6%	93.0%	96.3%	92.2%
twonorm	7400	80.6%	97.7%	97.2%	97.9%	93.1%
yeast	1484	47.7%	44.1%	47.1%	57.1%	53.0%
Average	NA	**76.1%**	**69.6%**	**78.5%**	**83.3%**	**78.4%**

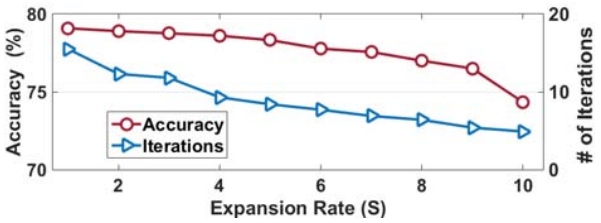

Fig. 6. Impact of expansion rate on SemiHD classification accuracy and the number of required iterations.

C. Expansion Rate

Figure 6 shows the impact of the expansion rate on the SemiHD classification accuracy and the number of required iterations to converge. The results are the average values on 18 tested datasets. Our evaluation shows that increasing the expansion rate from 5% slightly degrades SemiHD classification accuracy. SemiHD using a large expansion rate adds many data points to the labeled data which we are not sure about the correctness of their labels. This can cause a large fluctuation on the classification accuracy during retraining which results in a possible divergence. For example, we observe that 55% of the tested applications diverge when the expansion rate is equal to or larger than 35%. From another hand, using a larger expansion rate accelerates the training by reducing the number of required retraining iterations. Using a large expansion rate is equivalent to using a large learning rate. SemiHD uses an expansion rate to provide a trade-off between the classification accuracy and the training speedup. Depending on the user requirement, SemiHD can use a large expansion rate to provide fast training with acceptable classification accuracy or use small rate for high accuracy but relatively slow training.

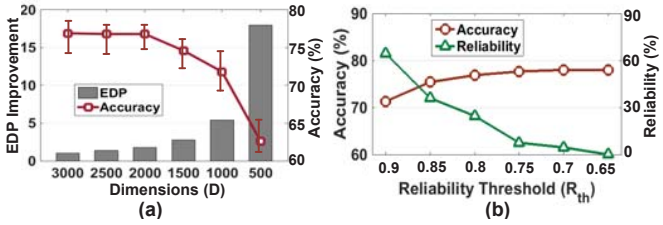

Fig. 7. Impact of (a) dimensionality on SemiHD accuracy and efficiency, (b) reliability threshold on SemiHD reliability.

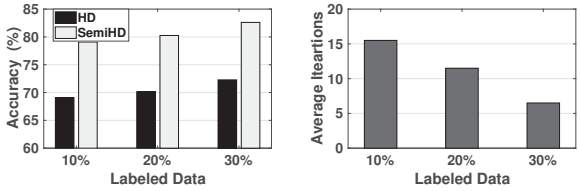

Fig. 8. Impact of the percentage of labeled data on maximum accuracy and number of required iterations.

D. Dimensionality & Reliability

Dimensionality: SemiHD can exploit hypervector dimensions as a parameter to trade efficiency and accuracy. Regardless of the dimension of the model at training, SemiHD can use a model in lower dimensions in order to accelerate the SemiHD inference. In HD computing the dimensions are independent, thus SemiHD can drop any arbitrary dimension in order to accelerate the computation. Figure 7 shows the classification accuracy and energy-delay product (EDP) improvement of SemiHD when the hypervector dimension changes from $D = 500$ to 3000. All EDP results are normalized to EDP of SemiHD with $D = 3000$ dimensions. Our evaluation shows that SemiHD can achieve maximum accuracy using dimensions equal to or larger than $D = 1500$. In addition, reducing dimension to $D = 500$ ($D = 1000$), SemiHD can achieve 17.9× and 5.4× higher EDP than SemiHD with $D = 3000$ while providing only 7.1% (2.5%) lower classification accuracy.

Reliability: To check the reliability of SemiHD model, we perform an experiment to show the response of the current HD model in classifying *out of scope* data. We selected random data points such that they are completely irrelevant to the existing classes. We have trained SemiHD model for random data such that the model learns to return "I don't know" whenever test data has less similarity with all class hypervectors. Figure 7b shows the impact of the reliability confidence, R_{th}, on the classification accuracy and reliability of SemiHD. The classification accuracy is defined as the percentage of data points (excluding random data) that are correctly classified by SemiHD. Similarly, the reliability of the model is defined as a portion of out of scope data points which can be correctly detected by SemiHD. Our evaluation shows that reliability confidence results in a trade-off between model accuracy and reliability. Using small confidence, e.g., $R_{th} = 0.75$, SemiHD provides high classification accuracy, but the model is not reliable to detect out of scope data points. Using large reliability confidence increases the quality loss in the main classifier since several data points will be incorrectly detected as out of scope data. However, this approach significantly improves the reliability of the model (as shown in Figure 7b). Depending on the application, our framework enables users to identify the importance of accuracy or model reliability. Our evaluation shows that using $R_{th} = 0.8$ provides the highest accuracy in identifying out of scope data. It should be noted that in order to provide more accurate results, the reliability confidence can be extracted individually for each application.

E. SemiHD & Percentage of Labeled

Figure 8 shows the impact of the percentage of the labeled data on SemiHD classification accuracy and the number of required retraining iterations. The results are the average values on all tested datasets. As we expected, SemiHD can achieve higher accuracy when classifying applications with larger labeled data. Applications with larger labeled data require fewer iterations to converge. Our evaluation shows that SemiHD using 30% labeled data can achieve, on average, 2.4% higher classification accuracy as compared to SemiHD using 10% labeled data. Figure 8 also compares the classification accuracy of SemiHD with the baseline HD that trained based on the labeled data. Our evaluations show that SemiHD can provide larger shift on the classification accuracy when using a small amount of labeled data. For example, using 10% labeled data, SemiHD can boost the HD classification accuracy by 10.2%, while the accuracy boost is only 8.3% on 30% labeled data.

F. SemiHD Acceleration

As we explained in Section VI-B, SemiHD using non-binary model provides significantly higher accuracy than the binary model. We reduce the dimensionality of the non-binary model to $D = 500$ such that it provides the same accuracy as the binary model with full dimensionality ($D = 3000$). Table II shows the energy consumption and execution time of SemiHD on FPGA and CPU when non-binary and binary models are using $D = 500$ and 3000 dimensions respectively. Here we observe that the selection of low dimensional non-binary or high-dimensional binary models depends on the underlying hardware, i.e., CPU or FPGA.

Let's consider SemiHD classification with two modules: encoder and classifier. Regardless of mapping data to the binary or non-binary domain, SemiHD spends a similar amount of energy and time for the encoding. The cost of the encoding module increases linearly with the number of dimensions. Therefore, the encoding module in the binary model has a higher cost than the non-binary model with lower dimension. From other hands, mapping to binary domain reduces the computation complexity of the classification task. In the binary domain, SemiHD can be trained and tested using the accumulation of the binary vectors and performing Hamming distance similarity. FPGA can accelerate training and inference computation using LUT resources. However, for a non-binary model, FPGA requires to use DSPs in order to perform training accumulation and associative search (dot product). Due to the limited number of DSPs, the efficiency of SemiHD with the non-binary model is bounded by the number of DSPs. In

TABLE II
EXECUTION TIME AND ENERGY CONSUMPTION OF DIFFERENT ALGORITHMS DURING TRAINING AND INFERENCE.

		CPU		FPGA	
		Non-binary	Binary	Non-binary	Binary
Training	*Exe.(s)*	4.43	5.34	5.22	0.35
	Energy(J)	13.67	11.95	9.36	1.04
Inference	*Exe.(ms)*	0.64	0.69	0.73	0.09
	Energy(mJ)	2.27	1.94	2.09	0.18

contrast, CPUs are not designed to work with very long sized binary vectors, thus they provide lower efficiency during the classification when using long binary vectors. We observe that CPU provides high efficiency when running non-binary low-dimensional vectors ($D = 500$), while FPGA is more efficient using high-dimensional binary vectors ($D = 3000$). Table II compares the efficiency of SemiHD using binary and non-binary models on both FPGA and CPU platforms. The results show that SemiHD using FPGA implementation ($D = 500$) can achieve $7.11\times$ faster and $12.6\times$ energy efficiency as compared to the best CPU implementation ($D = 3000$) which provides the same accuracy.

VII. CONCLUSION

In this paper, we propose SemiHD, a novel semi-supervised algorithm based on brain-inspired HD computing. SemiHD maps data points into high-dimensional space and trains a model based on the available labeled data. SemiHD iteratively expands the training data by labeling data points which can be classified by the current model with high confidence. We also proposed a framework which enables users to trade accuracy-efficiency along with the model reliability at runtime. Our evaluation shows that SemiHD can improve the classification accuracy of supervised HD by 10.2% and provide faster computation as compared to state-of-the-art semi-supervised approaches.

ACKNOWLEDGEMENTS

This work was partially supported by CRISP, one of six centers in JUMP, an SRC program sponsored by DARPA, Semiconductor Research Corporation (2016-TS-2690), and NSF grants 1527034, 1730158, 1911095, 1826967, and CNS-1619261.

REFERENCES

[1] J. Gantz and D. Reinsel, "The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east," *IDC iView: IDC Analyze the future*, vol. 2007, no. 2012, pp. 1–16, 2012.

[2] J. Konečnỳ *et al.*, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[3] B. Varghese *et al.*, "Challenges and opportunities in edge computing," *arXiv preprint arXiv:1609.01967*, 2016.

[4] M. Imani *et al.*, "Floatpim: In-memory acceleration of deep neural network training with high precision," in *ISCA*, pp. 802–815, ACM, 2019.

[5] J. Gubbi *et al.*, "Internet of things (iot): A vision, architectural elements, and future directions," *Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[6] I. Lee *et al.*, "The internet of things (iot): Applications, investments, and challenges for enterprises," *Business Horizons*, vol. 58, no. 4, pp. 431–440, 2015.

[7] D. Guinard *et al.*, "From the internet of things to the web of things: Resource-oriented architecture and best practices," in *Architecting the Internet of things*, pp. 97–129, Springer, 2011.

[8] 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.

[9] M. Imani *et al.*, "A framework for collaborative learning in secure high-dimensional space," in *Cloud Computing (CLOUD)*, IEEE, 2019.

[10] M. Imani *et al.*, "Sparsehd: Algorithm-hardware co-optimization for efficient high-dimensional computing," in *FCCM*, pp. 190–198, IEEE, 2019.

[11] S. Salamat *et al.*, "F5-hd: Fast flexible fpga-based framework for refreshing hyperdimensional computing," in *FPGA*, pp. 53–62, ACM, 2019.

[12] A. Rahimi, A. Tchouprina, P. Kanerva, J. d. R. Millán, and J. M. Rabaey, "Hyperdimensional computing for blind and one-shot classification of eeg error-related potentials," *Mobile Networks and Applications*, pp. 1–12, 2017.

[13] M. Imani *et al.*, "Bric: Locality-based encoding for energy-efficient brain-inspired hyperdimensional computing," in *DAC*, p. 52, ACM, 2019.

[14] C. Rosenberg *et al.*, "Semi-supervised self-training of object detection models.," in *WACV/MOTION*, pp. 29–36, 2005.

[15] D. P. Kingma *et al.*, "Semi-supervised learning with deep generative models," in *NIPS*, pp. 3581–3589, 2014.

[16] K. P. Bennett *et al.*, "Semi-supervised support vector machines," in *NIPS*, pp. 368–374, 1999.

[17] A. Subramanya *et al.*, "Graph-based semi-supervised learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 8, no. 4, pp. 1–125, 2014.

[18] I. Muslea *et al.*, "Active+ semi-supervised learning= robust multi-view learning," in *ICML*, vol. 2, pp. 435–442, Citeseer, 2002.

[19] M. Imani *et al.*, "Voicehd: Hyperdimensional computing for efficient speech recognition," in *ICRC*, pp. 1–8, IEEE, 2017.

[20] A. Rahimi *et al.*, "A robust and energy-efficient classifier using brain-inspired hyperdimensional computing," in *ISLPED*, pp. 64–69, ACM, 2016.

[21] M. Imani *et al.*, "Low-power sparse hyperdimensional encoder for language recognition," *IEEE Design & Test*, vol. 34, no. 6, pp. 94–101, 2017.

[22] M. Imani *et al.*, "Exploring hyperdimensional associative memory," in *HPCA*, pp. 445–456, IEEE, 2017.

[23] M. Imani *et al.*, "Hdna: Energy-efficient dna sequencing using hyperdimensional computing," in *BHI*, pp. 271–274, IEEE, 2018.

[24] Y. Kim *et al.*, "Efficient human activity recognition using hyperdimensional computing," in *IoT*, p. 38, ACM, 2018.

[25] M. Imani *et al.*, "Hdcluster: An accurate clustering using brain-inspired high-dimensional computing," in *DATE*, IEEE, 2019.

[26] M. Imani *et al.*, "A binary learning framework for hyperdimensional computing," in *DATE*, pp. 126–131, IEEE, 2019.

[27] M. Imani *et al.*, "Fach: Fpga-based acceleration of hyperdimensional computing by reducing computational complexity," in *ASP-DAC*, pp. 493–498, ACM, 2019.

[28] J. Morris *et al.*, "Comphd: Efficient hyperdimensional computing using model compression," in *ISLPED*, IEEE, 2019.

[29] S. Gupta *et al.*, "Felix: Fast and energy-efficient logic in memory," in *ICCAD*, pp. 1–7, IEEE, 2018.

[30] M. Imani *et al.*, "Hierarchical hyperdimensional computing for energy efficient classification," in *DAC*, p. 108, ACM, 2018.

[31] T. Feist, "Vivado design suite," *White Paper*, vol. 5, 2012.

[32] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in python," *JMLR*, vol. 12, pp. 2825–2830, 2011.

[33] I. Triguero *et al.*, "Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study," *Knowledge and Information systems*, vol. 42, no. 2, pp. 245–284, 2015.