RegHD: Robust and Efficient Regression in Hyper-Dimensional Learning System

Alejandro Hernández-Cano*, Cheng Zhuo$^{\psi}$, Xunzhao Yin$^{\psi}$, Mohsen Imani†⋆
*Universidad Nacional Autónoma de México, $^{\psi}$Zhejiang University, †University of California Irvine
⋆Email: m.imani@uci.edu

***Abstract*—Machine learning (ML) algorithms are key enablers to effectively assimilate and extract information from many generated data in the Internet of Things. However, running ML algorithms often results in extremely slow processing speed and high energy consumption. To achieve real-time performance with high energy efficiency and robustness, we proposed RegHD, the first regression solution based on Hyperdimensional computing. RegHD redesign a regression algorithm using strategies that more closely model the ultimate efficient learning machine: *the human brain*. RegHD performs regression after mapping data points into high-dimensional space using similarity preserving encoding. Due to the encoder's non-linearity, RegHD learns a regression model in an efficient and linear way. RegHD creates two set of models: *Input Model* to cluster data points with high similarity, and *Regression Model* to generate a regression model for each clustered data. During prediction, RegHD computes the output value by the weighted accumulation of all regression models, considering the model confidence obtained during similarity search. To improve RegHD efficiency, we also proposed a framework that enables RegHD model quantization while having no impact on the learning accuracy. Our evaluation shows that RegHD provides 5.6× and 12.3× (2.9× and 4.2×) faster and energy efficient training (inference) as compared to state-of-the-art regression algorithms, while providing similar quality of learning.**

I. Introduction

Internet of Things generates a large amount of raw data that could be entirely meaningless unless they process by Machine Learning (ML) algorithms [1], [2]. Regression is the key learning algorithm widely used for prediction, forecasting, and causal relationships between variables. Besides, regression is the main building block to enable accurate reinforcement learning. However, running machine learning (ML) algorithms often results in extremely slow processing speed and high energy consumption on traditional systems, or needs a large cluster of application-specific integrated chips (ASIC), e.g., Google TPU [3]–[5], and emerging hardware [6]. In addition, ML algorithms in the training phase have very high sensitivity to noise and failure in the hardware [7]–[10].

The human brain can do much of these learning effortlessly [11]–[15]. Hyperdimensional (HD) computing is introduced as an alternative computational model mimicking "the human brain" in the functionality level. HD computing is based on the fact that the brain works with neural activities in high-dimensional space. HD computing is well suited to address learning tasks for IoT systems as: (i) it is computationally efficient and highly parallel at heart to train and amenable to hardware level optimization [16], [17], (ii) it offers an intuitive and human-interpretable model [18], and (iii) it provides strong robustness to noise – a key strength for IoT systems. These features make HD computing a promising solution for today's embedded devices with limited storage, battery, and resources. Several prior work look at the application of HD computing for classification and clustering [19], [20]. To enable robust and real-time learning system, one possible solution is to exploit HD computing for regression. However, HD computing is an approximate computational model, and naively extending classification to perform regression results in low quality of learning and significant computational cost [18].

In this paper, we proposed RegHD, a novel approach to perform regression in high-dimensional space. RegHD fundamentally revisits HD algorithms in order to design a novel and efficient approach for regression. RegHD preserves parallelism, robustness, and efficiency of the HD-based system while ensuring high quality of regression. The main contributions of the paper are listed below:

- To the best of our knowledge, RegHD is the first regression algorithm based on hyperdimensional computing. RegHD exploits similarity preserving encoding and runtime clustering of data points in order to create multiple regression models. During training, RegHD creates two set of models: *Input Model* to cluster data points with high similarity, and *Regression Model* to perform the prediction. During prediction, RegHD computes the output value by the weighted accumulation of all regression models, depending on the similarity of input data to the cluster model.
- We propose a novel framework for quantizing the cluster and regression model during the training phase. To reduce the learning cost, RegHD reduces the precision of the cluster model to binary hypervectors, enabling RegHD to use hardware-friendly Hamming distance as a similarity metric. RegHD exploit a similar framework for binarizing data points and regression model with minimal impact on the regression accuracy. This enables RegHD to enable real-time learning by trading accuracy and efficiency.
- We evaluate RegHD efficiency on a wide range of regression problems and compare the results with the state-of-the-art regression algorithms. Our evaluation shows that RegHD provides comparable accuracy to state-of-the-art learning systems while provides significantly high computation efficiency. For example, RegHD trains 5.6× faster and is 12.3× more energy efficiency as compared to tested neural networks.

II. RegHD: Hyperdimensional Regression

A. Overview

Figure 1 shows an overview of Hyperdimensional computing performing regression. The first step in HD computing is to map each data points into high-dimensional space. The mapping procedure is often referred to as *encoding*. During regression, RegHD creates two set of models: *Input Model* to cluster data points with high similarity, and *Regression Model* to perform the prediction. Each model consists of multiple vectors with the same dimensionality as encoded data points. Each vector in the regression model corresponds to a cluster of inputs aggregated in a cluster hypervector. During training,

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

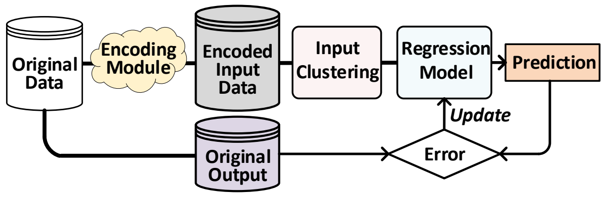

Fig. 1. RegHD overview performing regression.

RegHD first checks the similarity of a data point with the input model. Depending on the search result, RegHD (i) adds encoded data with the corresponding cluster vector with the highest similarity, and (ii) updates corresponding vectors in the regression model. The training performs iteratively over training data until RegHD learns a suitable model. During prediction, RegHD uses the same encoding module to map each query data into high-dimensional space. RegHD checks the similarity of the encoded data with the input model and uses corresponding vectors in the regression model for prediction. The prediction is a dot product between encoded data and model hypervectors. In the rest of the section, we explain the details of RegHD functionality in each step.

B. Encoding

HD computing uses different encoding methods depending on data types [21], [22]. The encoded data should satisfy the common-sense principle: data points that are different from each other in the original space should also be different in the HD space. For example, if a data point is completely different from another one, the corresponding hypervectors should be orthogonal in the HD space. Here, we show the encoding module for a feature vector, which is one of the complex data representations. In the feature vector, we do not know the relation between different features; thus, the encoding module should find out the importance of the features and the relation between them. 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, $\vec{\mathcal{H}} = (h_1,\ h_2, \cdots, h_n) \in \mathcal{R}^D$, where $D \gg n$. The following equation shows an encoding method that maps input vector into high-dimensional space:

$$h_i = \cos(\vec{F} \cdot \vec{\mathcal{B}}_i + b_i)\sin(\vec{F} \cdot \vec{\mathcal{B}}_i) \qquad (1)$$

where $\vec{\mathcal{B}}_k$s are randomly chosen hence orthogonal base hypervectors of dimension $\mathcal{D} \simeq 10k$ to retain the spatial or temporal location of features in an input and $b_i \sim \mathcal{U}(0, 2\pi)$. That is, $\vec{\mathcal{B}}_{kj} \sim \mathcal{N}(0,1)$ and $\delta(\vec{\mathcal{B}}_{k_1}, \vec{\mathcal{B}}_{k_2}) \simeq 0$, where δ denotes the cosine similarity.

C. Single-Model Regression

Let us assume the pairs of (x, y) as a sample of training data points in the original space. The main goal of regression is to learn the function between x and y, given all training data points. For any given x, the trained model needs to correctly predict y. Figure 2a shows RegHD functionality using single-model regression. RegHD creates a model to perform this prediction ($\vec{\mathcal{M}}$). Initially, this model is a hypervector initialized to all zero elements, $\vec{\mathcal{M}} \in \{0\}^D$. For every training data (x, y), RegHD first encodes the input data into high-dimensional space, $\vec{\mathcal{S}} \in \{-1, +1\}^D$. The dot product operation between the encoded input hypervector and the model predicts the output value: $\hat{y} = \vec{\mathcal{M}}.\vec{\mathcal{S}}$, where $\hat{y}$ is a scalar value. The difference of the predicted value ($\hat{y}$) and the correct output (y) represents RegHD prediction error. RegHD updates the model depending on the error of each prediction as follows:

$$\vec{\mathcal{M}} \leftarrow \vec{\mathcal{M}} + \alpha(y - \vec{\mathcal{M}}.\vec{\mathcal{S}}) \times \vec{\mathcal{S}} \qquad (2)$$

where α is a learning rate and $E = y - \vec{\mathcal{M}}.\vec{\mathcal{S}}$ represents the prediction error . The model update adds input data to the model, depending on the prediction error.

RegHD creates a single-pass model by one time going though all training data points and updating the model. However, this model often provides low accuracy as the last inputs have a higher chance of modifying the model toward their desired direction. To address this issue, RegHD performs iterative learning where we look at training data points iterative and update the model. The model retraining stops when RegHD has minor changes on the model M during a few consecutive iterations. During prediction phase, RegHD predict an output value for a query $\vec{\mathcal{X}}$ using the following steps: (i) first encode data point into high-dimensional space ($\vec{\mathcal{Q}}$) using the same encoding module used during training, (ii) RegHD predicts output value using: $\hat{y} = \vec{\mathcal{Q}}.\vec{\mathcal{M}}$

Figure 3a shows the regression accuracy during different retraining iterations. Our evaluation indicates that improves the model during iterative learning process, meaning that RegHD predicted value will be closer to the actual output. Figure 3b also shows RegHD regression for more complex tasks. As results indicate, RegHD has a limitation on learning a suitable regression model on a sophisticated dataset. In the following, we discuss the limitations of RegHD single-model.

Hypervector Capacity: Depending on the hypervector dimensionality, a single hypervector has limited capacity to store information. The accumulation of the encoded inputs during model training can result in the saturation of the model hypervector. In fact, the input with a more common pattern will dominate the model, results in losing the information of several less frequent inputs.

Let us assume that $\vec{\mathcal{M}}$ is the addition of P hypervectors (i.e., P distinct patterns), $\vec{\mathcal{M}} = \vec{\mathcal{S}}_1 + \cdots + \vec{\mathcal{S}}_P$. To check, if vector $\vec{\mathcal{M}}$ store the pattern of all input data, we perform the following dot product:

$$\delta(\vec{\mathcal{M}}, \vec{\mathcal{Q}}) = \delta(\vec{\mathcal{S}}_\lambda, \vec{\mathcal{Q}}) + \sum_{i=1, i\neq\lambda}^{P} \delta(\vec{\mathcal{S}}_i, \vec{\mathcal{Q}}) \qquad (3)$$

If $\vec{\mathcal{H}}_\lambda = \vec{\mathcal{Q}}$, thus $\delta(\vec{\mathcal{H}}_\lambda, \vec{\mathcal{Q}}) = D$ and the noise terms is nearly zero, as $\vec{\mathcal{S}}$ vectors are random vectors with nearly orthogonal distribution. In order to identify if $\vec{\mathcal{M}}$ already stored the $\vec{\mathcal{Q}}$ information, $\frac{\delta(\vec{\mathcal{M}}, \vec{\mathcal{Q}})}{D} > T$. For example, using $D = 100,000$ and $T = 0.5$, we can identify $P = 10,000$ patterns with 5.7% error. Although, using larger D values can increase the capacity of $\vec{\mathcal{M}}$, this comes with the cost of more resources.

D. Multi-Model Regression

The limited capacity of $\vec{\mathcal{M}}$ hypervector and the simplicity of the predictions, eliminates RegHD capacity to provide suitable regression results on complex regression problems. To address this issue, we RegHD introduce the idea of multi-model regression. RegHD exploits multiple hypervectors to store the information of data points with different patterns. As Figure 2b shows, the main idea behind this approach is to cluster input data and create a separate model for each input cluster. During inference, RegHD first checks to which of the

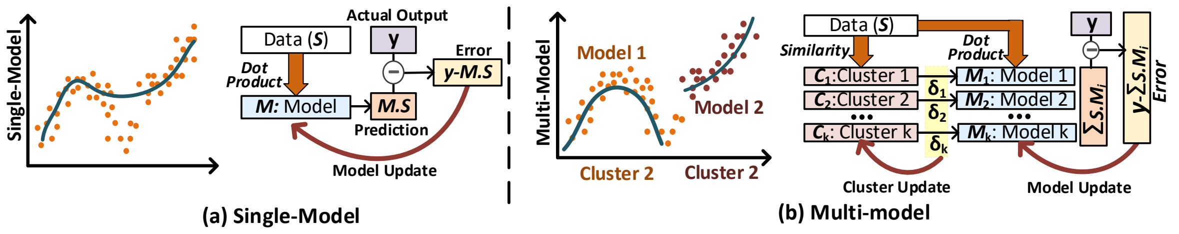

Fig. 2. RegHD functionality using (a) single-model and (b) multi-model regression.

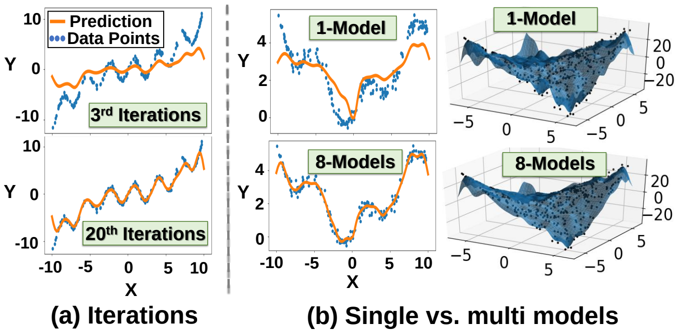

Fig. 3. RegHD quality of regression during (a) different training iterations, (b) using single and multi-vector models.

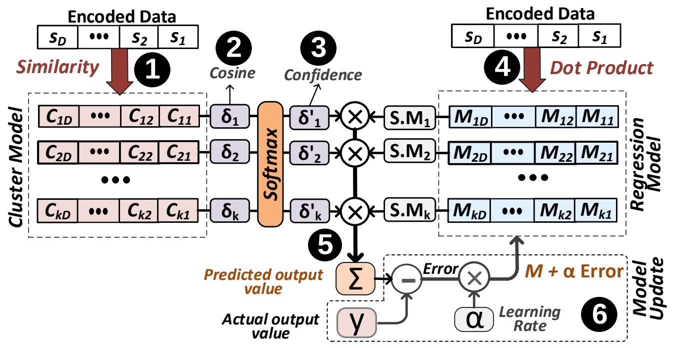

Fig. 4. RegHD training and prediction using multi-model.

cluster centers a data point is assigned to, then accordingly uses the selected model to perform the regression task. In other words, RegHD multi-model learning performs clustering and regression at the same time. This enables RegHD to have a more accurate model of a learned function.

Initial Model generation: Let us assume RegHD with k models. RegHD stores two sets of hypervectors: cluster hypervectors ($\{\vec{\mathcal{C}}_1, \vec{\mathcal{C}}_2, \cdots, \vec{\mathcal{C}}_k\}$) and model hypervectors ($\{\vec{\mathcal{M}}_1, \vec{\mathcal{M}}_2, \cdots, \vec{\mathcal{M}}_k\}$). The cluster hypervectors are initialized to random binary values, while model hypervectors are initialized as zero hypervectors. For a pair of (x, y), RegHD encodes inputs into high-dimensional space, $\vec{\mathcal{S}}$.

Model Update: Figure 4 shows RegHD functionality using multi regression models. RegHD checks the similarity of $\vec{\mathcal{S}}$ with all cluster hypervectors (❶). Each similarity value shows the confidence that a data point belongs to that cluster. For i^{th} cluster, the cosine similarity is computed as(❷). Next, RegHD normalizes the similarity values (δ) by passing them through a normalization block, e.g., softmax. The $\delta'(S, C_i)$ indicates the confidence of each cluster (❸). For an encoded hypervector $\vec{\mathcal{S}}$, RegHD predicts the output value using all models and their confidence value(❹): $\hat{y} = \sum_{i=1}^{k} \delta'(S, \vec{\mathcal{C}}_i)\ \vec{\mathcal{M}}_i.\vec{\mathcal{S}}$.

The predicted value is the weighted accumulation of all regression models (❺). The weight of each model, $\delta'(S, C_i)$, determines the confidence of each cluster center for having S. During training, RegHD updates the model based on how far is this prediction from the actual output value (❻):

$$\vec{\mathcal{M}}_i \leftarrow \vec{\mathcal{M}}_i + \alpha\underbrace{(y - \hat{y})}_{Error} \times \vec{\mathcal{S}} \qquad (4)$$

where term '$y - \hat{y}$' indicates the error between the actual output and predicted result and 'α' is the learning rate. RegHD also updates the cluster centers by addining input data to a center that has the highest cosine similarity. For example, if an input data has maximum similarity with l center, the cluster center updates as follows: $\vec{\mathcal{C}}_l = \vec{\mathcal{C}}_l + (1 - \delta_l) \times \vec{\mathcal{S}}$.

The term '$1 - \delta_l$' ensures that we do not saturate the cluster hypervectors with dominant input patterns. If an input data already exists in the $\vec{\mathcal{C}}_l$, RegHD will update the cluster center with small weight, as $1 - \delta_l \simeq 0$. This update is equivalent to clustering input data and ensuring that we only select suitable model hypervector for cluster update. After updating the cluster and regression models, RegHD continues with an iterative update over training data points until the quality of regression stabilizes during the last few iterations. This iterative process ensures that cluster centers well-represent training data, and each prediction is assigned to a proper model based on the input distribution.

Prediction: The first step encodes the input to produce a query hypervector $\vec{\mathcal{S}}$. RegHD computes the similarity of encoded input with all cluster centers; then, it normalizes the similarity values to find each model's confidence. Finally, RegHD performs the prediction.

III. RegHD Efficient Implementation

As a light-weight learning, RegHD needs to enable real-time and reliable learning on today's IoT systems. RegHD requires to have the robustness to noise and failure on embedded devices which are working based on unreliable battery sources. More importantly, RegHD requires hardware efficiency as its intended to run on embedded devices with limited resources. In terms of robustness, RegHD has redundant representation, which provides inherent robustness to possible noise in the hardware. In RegHD, hypervectors store information across all their 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.

However, the algorithm explained in Section II, are not efficient enough to process on embedded processors with sub 1-watt power consumption. RegHD computation involves mainly similarity search on cluster hypervectors and dot product operations over model hypervector. All these operations are happening over hypervectors with integer values. Therefore, RegHD requires to sue costly cosine similarity and high-precision dot product operations. In this section, we propose

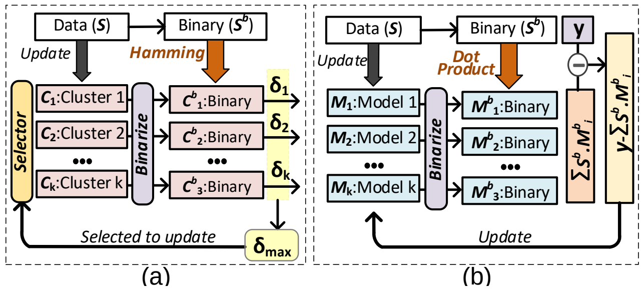

Fig. 5. Framework for RegHD model quantization.

a framework for binarizing the computation of RegHD with no or minimal impact on accuracy.

A. *Quantized Clustering*

One of the main computational cost of RegHD is the similarity search of encoded data with all cluster centers. Using cluster centers with integer representation, RegHD requires several high-precision arithmetic operations (multiplication and addition) to compute cosine similarity. Figure 5a shows an overview of the proposed framework. Our framework exploits cluster hypervectors with binary representation and uses efficient Hamming distance for similarity search. However, naive binarization of the cluster hypervectors results in a significant loss in the regression accuracy, as binary vectors do not have the capability for the model update.

RegHD stores two copy of the cluster hypervectors: integer ($\vec{\mathcal{C}}$) and binary ($\vec{\mathcal{C}}^b$) versions. Similarly, our encoding module maps each input data into hypervectors with integer ($\vec{\mathcal{S}}$) and binary ($\vec{\mathcal{S}}^b$) representation. During regression, RegHD checks the similarity of the binary data with the binary clusters ($\delta(\vec{\mathcal{C}}^b, \vec{\mathcal{S}})$). This similarity performs using efficient Hamming distance similarity. Depending on the similarity results, RegHD updates the model and cluster hypervectors. This update happens over integer clusters using integer input, as shown here: $\vec{\mathcal{C}_l} = \vec{\mathcal{C}_l} + (1 - \delta(\vec{\mathcal{C}}^b_l, \vec{\mathcal{S}}^b)) \times \vec{\mathcal{S}}$.

Update on integer vector ensures that the cluster has enough capacity (defined in Section II-C) to store information of multiple patterns. The binary search and integer update continue over all training data points (or batch of data). Next, RegHD updates binary clusters by quantizing the integer model. This quantization assigns each element of cluster hypervector to 0 or 1 by exploiting a single comparison operation. For the next iterations, RegHD uses an updated binary cluster for the rest of the regression task.

Figure 6 compares the quality of regression in RegHD using binary and integer cluster centers. The results are also compared to RegHD using naive binarization. Our evaluation indicates that our framework enables RegHD to provide similar regression quality as RegHD with integer model. This quality is significantly higher than RegHD using naive binarization. This is because our framework teaches RegHD to adapt itself to work with the binarization constraint during the iterative learning process. It should be noted that our framework comes with a slight increase in the number of training iterations. However, this overhead is negligible as RegHD speeds up each training iteration by eliminating costly cosine similarity.

B. *Quantized Prediction*

RegHD can exploit a similar framework to binarize model hypervectors. Figure 5b shows an overview of our framework.

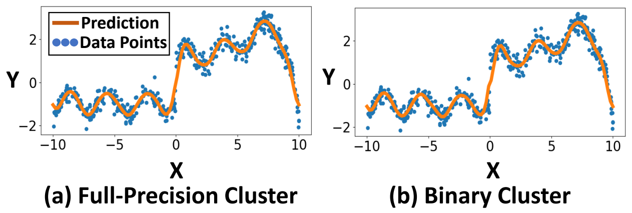

Fig. 6. RegHD regression w/wo cluster quantization.

This binarization aims to simplify the dot product operation between encoded data and the model to Hamming computing. To this end, RegHD stores two copies of the model: integer model ($\vec{\mathcal{M}}$) and binary model ($\vec{\mathcal{M}}^b$). RegHD has three choices for reducing the complexity of dot product operation.

- **Binary Query - Binary Model:** RegHD predicts the regression output by performing bitwise AND operation binary query and binary model hypervectors: $\hat{y} = \sum_{i=1}^{k} \delta' \ \vec{\mathcal{M}}^b_i . \vec{\mathcal{Q}}^b$
- **Binary Query - Integer Model:** RegHD predicts the regression output by performing dot product operation between binary query and integer model hypervectors: $\hat{y} = \sum_{i=1}^{k} \delta' \ \vec{\mathcal{M}}_i . \vec{\mathcal{Q}}^b$. In this configuration, the dot product is multiply-free and efficient as the query vector is binary.
- **Integer Query - Binary Model:** RegHD predicts the regression output by performing dot product operation between integer query and binary model: $\hat{y} = \sum_{i=1}^{k} \delta' \ \vec{\mathcal{M}}^b_i . \vec{\mathcal{Q}}$. This configuration also supports fast and efficient dot product.

Regardless of using any of the above predictions, during training, RegHD updates the integer model ($\vec{\mathcal{M}}_i \leftarrow \vec{\mathcal{M}}_i + \alpha(y - \hat{y}) \times \vec{\mathcal{S}}$). This is because the precision of the model update has an important impact on RegHD convergence and the final quality of the model. After going through all training data (or a batch), RegHD binarizes the model (if necessary) and use the new binary model for the next iteration of training. Depending on the prediction method, RegHD provides different accuracy and efficiency trade-offs.

IV. Evaluation

A. *Experimental Setup*

We implement RegHD using both software and hardware support. In software, we verified RegHD functionality using C++ implementation. In hardware, We implement RegHD training and testing on two embedded platforms: Kintex-7 FPGA and ARM Cortex A53 CPU. For FPGA, we design the RegHD functionality using Verilog and synthesize it using Xilinx Vivado Design Suite [23]. For CPU, the RegHD code has been written in C++ and optimized for performance. The code has been implemented on Raspberry Pi (RPi) 3B+. We evaluate RegHD accuracy and efficiency on popular regression datasets, including diabetes [24], Boston housing [25], NASA airfoil self-noise [26], wine quality prediction [27], Facebook performance metrics [28], combined cycle power plant (CCPP) prediction [29], and forest fire prediction [30].

B. RegHD *Quality*

State-of-the-art: We compare RegHD regression quality with state-of-the-art regression algorithms, including Deep Neural Network (DNN), Support Vector Regression (SVR), and Decision Tree. The DNN models are trained with Tensorflow [31], and we exploited the Scikit-learn library [32] for the other algorithms. Our evaluation shows that RegHD

TABLE I
COMPARING RegHD QUALITY OF REGRESSION (MEAN SQUARE ERROR) WITH THE STATE-OF-THE-ART REGRESSION ALGORITHMS.

	diabetes	*boston*	*airfoil*	*wine*	*facebook*	*CCPP*	*forest*
DNN	3385.1	14.6	24.2	0.51	11890.7	19.9	701.2
Decision Tree	5508.7	32.3	19.0	0.59	11740.9	22.8	1437.5
SVR	4756.2	13.5	16.7	0.64	13821.5	24.3	1104.7
Baseline-HD [18]	7921.3	62.3	48.9	73.10	18340.0	51.3	1775.2
RegHD-1	5658.8	23.0	20.4	0.61	12933.1	23.6	906.9
RegHD-2	4981.7	20.5	18.2	0.54	11798.6	21.2	807.8
RegHD-8	4336.6	17.5	17.0	0.53	11344.8	20.2	762.1
RegHD-32	3982.2	15.8	16.0	0.53	11117.9	20.0	746.9

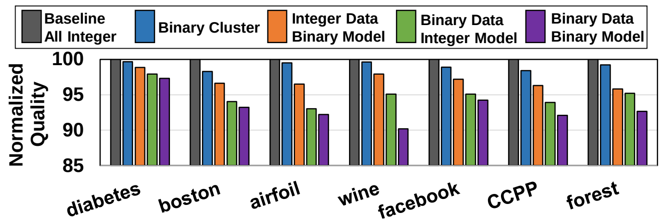

Fig. 7. RegHD quality in different configurations.

provides comparable quality of regression to other algorithms. Table I compares RegHD quality of regression with state-of-the-art HD-based algorithms [18], called *Baseline-HD*. The baseline-HD [18] emulates the regression task by exploiting HD classification with several hypervectors. Each hypervector represents a range of output. Depending on the search result, RegHD decides to select an output with the highest similarity as a prediction result. Our evaluation shows that the baseline-HD provides a significantly low quality of regression, especially on high-precision applications. In contrast, RegHD natively supports regression using iterative training, enabling us to work with a minimal number of hypervectors. Table I also lists the impact of the number of vectors in RegHD classification accuracy. RegHD with more number models improves the regression accuracy. For example, RegHD with 32-models (RegHD-32) provides, on average, 21.3% higher quality of regression. This accuracy improvement comes at the cost of lower computation efficiency.

Configurations: Figure 7 shows RegHD normalized quality of regression using quantized clustering and quantized model. Our evaluation indicates that RegHD achieves maximum quality using a quantized cluster (only 0.3% lower). This is because the cluster model does not have a direct impact on the final prediction result. In contrast, model quantization can add a large amount of error to the regression result. Using a binary query - binary model results in a maximum quality loss, as model prediction is really approximated. RegHD with integer input - binary model also provides a relatively low quality of regression, on average, 5.2% lower quality than full precision RegHD. However, RegHD using binary query and integer model provides very similar regression result as the full precision RegHD (only 1.5% lower). This suggests that binarizing query has a lower impact on regression quality as compared to model binarization.

C. RegHD *Efficiency*

State-of-the-art: Figure 8 compares RegHD efficiency with DNN and the baseline HD during training and inference phase on Xilinx Kintex-7 FPGA. All results are reported RegHD using a binary cluster. For DNN, we used DNNWeaver V2.0 [33] for inference and FPDeep [34] for training implementation on a single FPGA device. FPGA implementations are optimized to maximize performance by utilizing

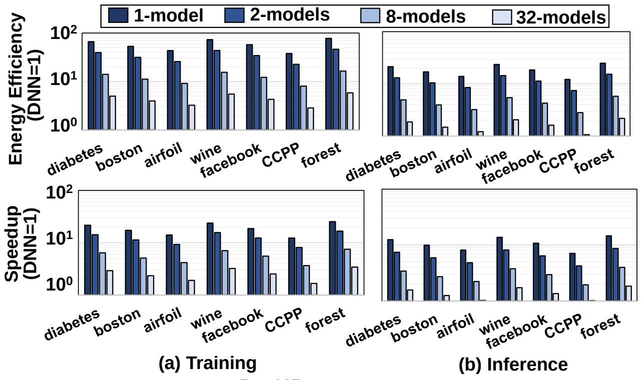

Fig. 8. Comparing RegHD training and inference efficiency.

the resources. Our evaluation shows that RegHD provides higher efficiency than DNN in both training and inference phases. During training, RegHD efficiency comes from (i) reducing the number of training iterations and (ii) improving the efficiency of a single training iteration by eliminating costly gradient operations required by DNN. During inference, RegHD efficiency is becoming closer to DNN, as DNN already supports fast single-pass inference with no costly iteration. Our evaluation shows that RegHD using 8-models provides 5.6× (2.9×) faster and 12.3× (4.2×) higher energy efficiency as compared to DNN during training (inference). As Figure 8 shows, RegHD efficiency also depends on the number of hypervectors used for cluster and regression model. Increasing the number of hypervectors linearly increases RegHD computation cost. RegHD with 8-models (2-models) provides 2.8× and 2.1× (4.9× and 8.0×) faster and more energy efficient training as compared to RegHD with 32-models.

Configurations: Figure 9 compares the RegHD efficiency using different cluster and model quantization. Our evaluation indicates that clustering is taking a high cost of RegHD entire computation. Therefore, cluster quantization provides, on average, 1.9× faster and 2.1× higher energy efficiency than the baseline. Model quantization can also improve computation efficiency. As we predicted, RegHD with a binary query and binary model has maximum efficiency. Model binarization or query binarization are also providing high computation efficiency. Since RegHD with binary query has a higher quality of regression, we prefer this configuration. Our evaluation shows that RegHD with a binary query and integer model (binary model) provides, on average, 1.4× and 1.5× (1.6× and 1.8×) faster and more energy efficient training as compared to the baseline. Figure 9 also shows RegHD efficiency during the inference task. Quantized clusters and models have a higher impact on RegHD inference efficiency. This is because RegHD in the inference phase does not use cluster updates, which cannot be quantized anyway. RegHD using quantized clusters can achieve 2.0× faster and 2.3× higher energy efficiency at inference. Similarly, RegHD using model quantization (binary query, binary model) provides 1.5× faster and 1.6× more energy efficiency than the baseline.

D. RegHD *and Dimensionality*

Table II reports RegHD regression quality loss and efficiency when the size of hypervector increases from $D = 0.5k$ to $4k$. Our evaluation shows that RegHD has high robustness to reduce dimensionality. For example, RegHD using $D = 3k$ dimension provides a similar quality of regression as full

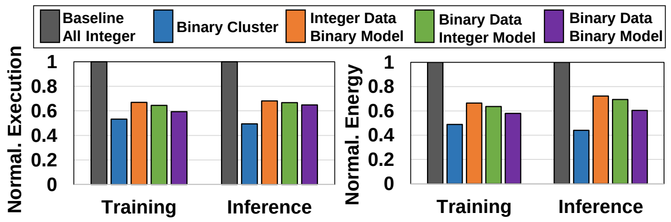

Fig. 9. RegHD efficiency in different configurations.

TABLE II
RegHD QUALITY LOSS AND EFFICIENCY IN DIMENSIONALITY

Dimensions		4k	3k	2k	1k	0.5k
Quality Loss		0%	0.1%	0.3%	0.9%	2.4%
Training	*Speedup*	1×	1.18×	1.71×	3.09×	5.20×
	Efficiency	1×	1.26×	1.86×	3.53×	6.38×
Inference	*Speedup*	1×	1.19×	1.78×	3.67×	7.13×
	Efficiency	1×	1.30×	1.90×	3.81×	7.62×

dimensionality. Further reducing dimensionality degrades the regression quality while improving the learning efficiency. Dimensionality reduction has a higher impact on inference efficiency. During training, reducing dimensionality results in increasing the number of required iterations. This reduces the linear improvement in energy consumption. For example, RegHD using $D = 1k$ provides 3.09× and 3.53× (3.67× and 3.81×) faster and higher energy efficiency during training (inference), while provides 0.9% lower regression accuracy.

V. Related Work

Since a computational neuroscientist P. Kanerva introduced the field of hyperdimensional computing [14], prior research have applied the idea into diverse cognitive tasks, such as latent semantic analysis, language recognition, gesture recognition, prediction from multimodal sensor fusion, and robotics [18], [35], [36]. Several prior works explored the capability of HD computing to enable single-pass training. For example, the work in [35] proposed a text classification algorithm based on random indexing as a scalable alternative to latent semantic analysis. Work in [18] used HD classification to mimic the regression function to predict the speed of a robot. However, this approach is a discrete (inaccurate) regression method that requires hundreds of class hypervectors, thus significantly inefficient in hardware. However, the application of existing HD algorithms is mainly in classification. RegHD is the firs algorithm that natively supports regression in high-dimensional space. RegHD exploits robustness and efficiency as key advantages that HD-based systems need to provide.

Prior work introduced iterative learning in HD classification. Work in [37] and [38] introduced iterative frameworks for quantizing and sparsifying the HD model during the training phase. Several recent works have presented the hardware accelerator for HD computing training and inference. This includes acceleration on the existing FPGAs [39]–[42] or designing new ASIC or processing in-memory architectures [16], [43]. Our approach is orthogonal to these hardware accelerators, as we can use these frameworks to sparsify the regression model and use similar platforms for RegHD acceleration.

VI. Conclusion

We propose RegHD, to the best of our knowledge, the first regression algorithm based on hyperdimensional computing. RegHD performs regression after mapping data into high-dimensional space using similarity preserving encoding. Due to the non-linearity of the encoder, RegHD learns a regression model efficiently and linearly. During prediction, RegHD computes the output value by the weighted accumulation of all regression models, considering the model confidence obtained during similarity search. To improve RegHD efficiency, we also proposed a framework that enables RegHD model quantization while having no impact on the learning accuracy.

Acknowledgment

This work was partially supported by Semiconductor Research Corporation (SRC) Task No. 2988.001 and Department of the Navy, Office of Naval Research, grant #N00014-21-1-2225.

References

[1] X. Wang *et al.*, "In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning," *arXiv preprint arXiv:1809.07857*, 2018.
[2] R. Marculescu *et al.*, "Edge ai: Systems design and ml for iot data analytics," in *KDD*, pp. 3565–3566, 2020.
[3] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *ISCA*, pp. 1–12, IEEE, 2017.
[4] I. Magaki *et al.*, "Asic clouds: Specializing the datacenter," in *ISCA*, pp. 178–190, IEEE, 2016.
[5] R. Andri *et al.*, "Yodann: An architecture for ultralow power binary-weight cnn acceleration," *TCAD*, vol. 37, no. 1, pp. 48–60, 2017.
[6] S. Angizi *et al.*, "Accelerating deep neural networks in processing-in-memory platforms: Analog or digital approach?," in *IVLSI*, pp. 197–202, IEEE, 2019.
[7] M. Courbariaux, Y. Bengio, and J.-P. David, "Training deep neural networks with low precision multiplications," *arXiv preprint arXiv:1412.7024*, 2014.
[8] S. Han *et al.*, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.
[9] Z. He *et al.*, "Sparse bd-net: a multiplication-less dnn with sparse binarized depth-wise separable convolution," *JETC*, vol. 16, no. 2, pp. 1–24, 2020.
[10] R. P. Bastos *et al.*, "Effects of transient faults in integrated circuits," in *Springer*, pp. 1–16, Springer, 2020.
[11] W. Zhang *et al.*, "Neuro-inspired computing chips," *Nature Electronics*, vol. 3, no. 7, pp. 371–382, 2020.
[12] J. Liu *et al.*, "Self-repairing learning rule for spiking astrocyte-neuron networks," in *ICONIP*, pp. 384–392, Springer, 2017.
[13] C. S. Thakur and thers, "Large-scale neuromorphic spiking array processors: A quest to mimic the brain," *Frontiers in neuroscience*, vol. 12, p. 891, 2018.
[14] 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.
[15] A. Calimera *et al.*, "The human brain project and neuromorphic computing," *Functional neurology*, vol. 28, no. 3, p. 191, 2013.
[16] M. Imani *et al.*, "Exploring hyperdimensional associative memory," in *HPCA*, pp. 445–456, IEEE, 2017.
[17] T. F. Wu *et al.*, "Brain-inspired computing exploiting carbon nanotube fets and resistive ram: Hyperdimensional computing case study," in *ISSCC*, pp. 492–494, IEEE, 2018.
[18] A. Mitrokhin *et al.*, "Learning sensorimotor control with neuromorphic sensors: Toward hyperdimensional active perception," *Science Robotics*, 2019.
[19] M. Imani *et al.*, "A framework for collaborative learning in secure high-dimensional space," in *CLOUD*, pp. 435–446, IEEE, 2019.
[20] L. Ge and K. K. Parhi, "Classification using hyperdimensional computing: A review," *IEEE Circuits and Systems Magazine*, vol. 20, no. 2, pp. 30–47, 2020.
[21] A. Rahimi *et al.*, "A robust and energy-efficient classifier using brain-inspired hyperdimensional computing," in *ISLPED*, pp. 64–69, ACM, 2016.
[22] M. Imani *et al.*, "Searchd: A memory-centric hyperdimensional computing with stochastic training," *TCAD*, 2019.
[23] T. Feist, "Vivado design suite," *White Paper*, vol. 5, 2012.
[24] "Diabetes patient records." https://archive.ics.uci.edu/ml/datasets/diabetes.
[25] "Boston Housing Dataset." http://lib.stat.cmu.edu/datasets/boston.
[26] R. L. Gonzalez, *Neural networks for variational problems in engineering*. PhD thesis, Universitat Politècnica de Catalunya (UPC), 2009.
[27] P. Cortez *et al.*, "Modeling wine preferences by data mining from physicochemical properties," *Decision Support Systems*, vol. 47, no. 4, pp. 547–553, 2009.
[28] S. Moro *et al.*, "Predicting social media performance metrics and evaluation of the impact on brand building: A data mining approach," *JBR*, 2016.
[29] P. Tüfekci, "Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods," *JEPE*, 2014.
[30] P. a. Cortez, "A data mining approach to predict forest fires using meteorological data," 2007.
[31] M. Abadi *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.
[32] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2825–2830, 2011.
[33] H. Sharma *et al.*, "From high-level deep neural models to fpgas," in *MICRO*, p. 17, IEEE Press, 2016.
[34] T. Geng *et al.*, "Fpdeep: Acceleration and load balancing of cnn training on fpga clusters," in *FCCM*, pp. 81–84, IEEE, 2018.
[35] P. Kanerva *et al.*, "Random indexing of text samples for latent semantic analysis," in *CogSci*, vol. 1036, Citeseer, 2000.
[36] A. Joshi *et al.*, "Language geometry using random indexing," in *QI*, pp. 265–274, Springer, 2016.
[37] M. Imani *et al.*, "Sparsehd: Algorithm-hardware co-optimization for efficient high-dimensional computing," in *FCCM*, pp. 190–198, IEEE, 2019.
[38] M. Imani *et al.*, "Quanthd: A quantization framework for hyperdimensional computing," *TCAD*, 2019.
[39] M. Schmuck *et al.*, "Hardware optimizations of dense binary hyperdimensional computing: Rematerialization of hypervectors, binarized bundling, and combinational associative memory," *JETC*, vol. 15, no. 4, pp. 1–25, 2019.
[40] S. Salamat *et al.*, "F5-hd: Fast flexible fpga-based framework for refreshing hyperdimensional computing," in *FPGA*, pp. 53–62, 2019.
[41] M. Imani *et al.*, "Revisiting hyperdimensional learning for fpga and low-power architectures," in *HPCA*, IEEE, 2021.
[42] A. Hernandez-Cano *et al.*, "A framework for efficient and binary clustering in high-dimensional space," in *DATE*, IEEE, 2021.
[43] M. Imani *et al.*, "Dual: Acceleration of clustering algorithms using digital-based processing in-memory," in *MICRO*, pp. 356–371, IEEE, 2020.