AdaptHD: Adaptive Efficient Training for Brain-Inspired Hyperdimensional Computing

Mohsen Imani‡, Justin Morris‡, Samuel Bosch*, Helen Shu‡, Giovanni De Micheli*, Tajana Rosing‡
‡Computer Science and Engineering, UC San Diego, La Jolla, CA, USA
* Integrated Systems Laboratory, EPFL, Lausanne, Switzerland
{moimani, j1morris, hsshu, tajana}@ucsd.edu; {samuel.bosch, giovanni.demicheli}@epfl.ch

***Abstract*—Brain-inspired Hyperdimensional (HD) computing is a promising solution for energy-efficient classification. HD emulates cognition tasks by exploiting long-size vectors instead of working with numeric values used in contemporary processors. However, the existing HD computing algorithms have lack of controllability on the training iterations which often results in slow training or divergence. In this work, we propose AdaptHD, an adaptive learning approach based on HD computing to address the HD training issues. AdaptHD introduces the definition of learning rate in HD computing and proposes two approaches for adaptive training: iteration-dependent and data-dependent. In the iteration-dependent approach, AdaptHD uses a large learning rate to speedup the training procedure in the first iterations, and then adaptively reduces the learning rate depending on the slope of the error rate. In the data-dependent approach, AdaptHD changes the learning rate for each data point depending on how far off the data was misclassified. Our evaluations on a wide range of classification applications show that AdaptHD achieves 6.9× speedup and 6.3× energy efficiency improvement during training as compared to the state-of-the-art HD computing algorithm.**

I. Introduction

Hyperdimensional (HD) computing is a brain-inspired computational approach which can go a long way in addressing the energy bounds that plague deterministic computing [1]. HD computing is based on the understanding that brains compute with *patterns of neural activity* that are not readily associated with numbers. In fact, the brain's ability to calculate with numbers is feeble. However, we can model neural activity patterns with points in a high-dimensional space. When the dimensionality is in the thousands (e.g., D = 10,000), it is called hyperdimensional. Operations on hypervectors can be combined into interesting computational behaviors with unique features that make them robust and efficient. HD has been used in different domains, including analogy-based reasoning[2], language recognition[3], [4], prediction from multimodal sensor fusion[5], [6], speech recognition [7], and brain-computer interfaces[8]. HD computing supports *single-pass* training, where the training can happen by only going once through a training data [4]. However, in this work, we argue that HD using single-pass training provides significantly low accuracy on practical classification problems, e.g., speech recognition and face detection. For example, for face detection application [9], single-pass learning provides 40% lower classification accuracy than the model which has been retrained. One efficient way to improve the HD classification accuracy is performing perceptron-like iterative training. However, without a clear definition of learning rate the retraining process often diverge or may take long time to converge. This makes the HD computing less desirable, as low-cost training is essential for embedded devices with limited battery and resources.

In this paper, we propose AdaptHD, a novel adaptive retraining method for HD computing. For the first time, AdaptHD introduces the definition of learning rate in HD computing and retrains a model using two adaptive approaches: iteration-dependent and data-dependent. In the iteration-dependent approach, AdaptHD changes the learning rate based on the changes in the classification accuracy during retraining. In the first iterations, this approach uses a large learning rate to speedup the training procedure, and then adaptively reduces the learning rate depending on the slope of the error rate. A second approach is a data-dependent approach, where AdaptHD changes the learning rate for each data point, depending on how far off the data was misclassified. Finally, AdaptHD proposes a hybrid approach which updates the learning rate considering both iteration and data dependency. We have evaluated the efficiency of AdaptHD on a wide range of classification applications. Our evaluation shows that AdaptHD using iteration-dependent (data-dependent) approach can achieve 6.9× speedup and 6.3× energy efficiency improvement during training as compared to the state-of-the-art HD computing algorithm.

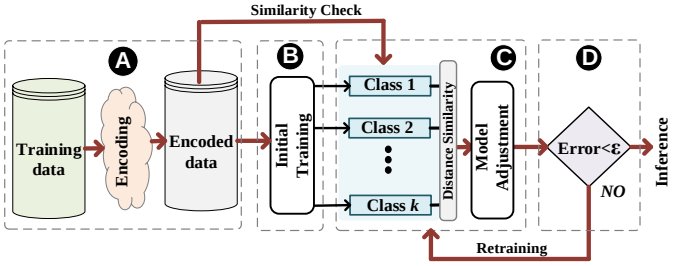

Fig. 1. Overview of a AdaptHD classification model.

II. AdaptHD Framework

A. HD Classification

In this section, we propose AdaptHD, an adaptive training framework for high-efficient and high-accurate Hyperdimensional computing. Figure 1 shows an overview of the AdaptHD performing the classification task. In AdaptHD, the encoder block maps input data into high-dimensional vectors, e.g., $D = 10,000$ dimensions, and combines them in order to generate a hypervector representing each class. During inference, the classifier performs the classification task by looking at the similarity of the input hypervector to each of the stored class hypervectors. In the following, we explain the functionality of each module.

Encoding: In HD computing, the first step is to map/encode all data points from original to a hypervector(❹). Here, we consider a general encoding approach which maps a feature vector $F = \{f_1, f_2, \ldots, f_n\}$, with n features ($f_i \in N$) to high-dimensional vector $Q = \{q_1, q_2, \ldots, q_D\}$ with D dimensions ($q_i \in \{0,1\}^D$) [1], [7]. This encoding finds the minimum 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 level $\{L_1, \ldots, L_m\}$. To preserve the feature position, 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 can happen by linearly combining the feature values over different indices, where the hypervectors corresponding to the feature indices preserve the position of each feature value in a combined set:

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

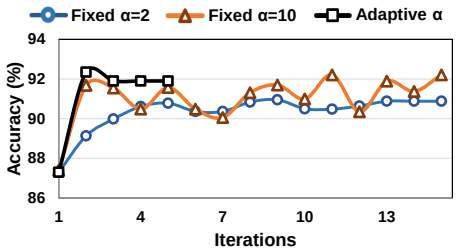

Fig. 2. HD classification accuracy using low and high learning rates.

where H is the (non-binary) encoded hypervector, $\oplus$ denotes the XOR operation, and $\overline{L}_i$ is the (binary) hypervector corresponding to the i-th feature of vector F. AdaptHD binarizes this hypervector by comparing each dimension of H with a threshold value, which is half of the number of features ($THR = n/2$).

Training: The encoded hypervectors are combined in a AdaptHD training module in order to create a single hypervector representing each class [4]. This combination is a simple addition of all encoded data points belong to a particular class. Then, the class hypervectors are binarized. For an application with k classes, AdaptHD results in generating k class hypervectors, $\{C_1, C_2, \ldots, C_k\}$ where $C_i \in \{0,1\}^D$. This training can happen by a single time passing through a training dataset (❸). This single-pass model provides good classification accuracy in a short amount of training time, but in practical applications, this accuracy is significantly lower than state-of-the-art (as explained in Section I).

Inference: In order to classify the input data, AdaptHD uses the same encoding module to map a test data into a high-dimensional vector, called *query hypervector*. AdaptHD calculates the Hamming distance similarity of each class hypervector with a query hypervector. Finally, a data point is assigned to a class which it has the highest similarity with it.

B. AdaptHD Learning Rate

To improve the HD model, we use a retraining approach which adjusts the class hypervectors by iterating over the training dataset (❹). After the initial training, AdaptHD saves a non-binary training model, $C_i \in N^D$. AdaptHD starts retraining by iteratively checking the similarity of each training data point with all class hypervector. Since the class hypervectors are non-binary, this similarity check happens using cosine metric. Our exportation is that all training data points should be correctly classified by the model. In case of a correct classification, we do not make any changes on the model. However, if Q_i is misclassified, we subtract Q_i from the incorrect class and add it to a class that it should have been matched with it (the label of the Q_i data). For the first time, AdaptHD introduces the definition of learning rate in HD computing. Learning rate, α, specifies the amount of changes that we make into a model during each retraining iteration. For example, for each misclassified data during retraining iteration, AdaptHD updates correct and incorrect class hypervectors using: $C_{wrong} = C_{wrong} - \alpha Q_i$ and $C_{correct} = C_{correct} + \alpha Q_i$. This process continues iteratively through training dataset until the algorithm converges (❺). The convergence defines as a time that the HD classification accuracy in the last three iterations changes less than 0.1%. After the convergence, AdaptHD binarizes the retrained model in order to enable the inference task to be performed using as efficient Hamming distance similarity check.

Figure 2 shows the impact of using different learning rates on the accuracy of speech recognition application [10]. Our evaluation shows that using a small learning rate, e.g., $\alpha = 2$, HD requires many retraining iterations to get near the best accuracy. Lower learning rates are better for fine-tuning the model towards the end because the accuracy does not fluctuate as much. On the other hand, with higher learning rates, we can get near the best accuracy faster. Higher learning rates are not suitable for fine-tuning the model once we are near the best accuracy. In fact, large learning rate results in the accuracy fluctuation which increases the possibility of divergence (as depicted in Figure 2). Often time HD using high learning rates do not achieve as high of an accuracy as lower learning rates. This is also due to the fact the lower learning rates are better at fine-tuning the HD model once it is near the best accuracy it can achieve.

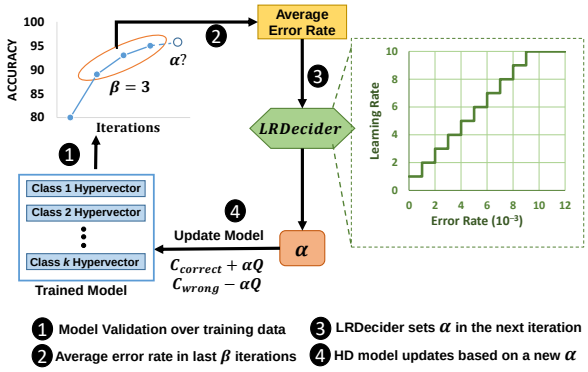

Fig. 3. AdaptHD framework supporting iteration-dependent training.

When adaptively changing α, we want to take advantage of higher learning rates ($\alpha >>$) to get near the best accuracy faster. Once we are near the best accuracy, we want to take advantage of lower learning ($\alpha <<$) rates to fine-tune the model. In this work, AdaptHD goal is to get the advantage of both small and larger learning rates in order to provide high classification accuracy as well as fast training. The black line in Figure 2 shows the AdaptHD classification accuracy when using the adaptive learning rate. This approach uses a large learning rate in the first iterations of training in order to accelerate the training process but eventually reduces the learning rate in order to converge.

C. AdaptHD Iterative Training

AdaptHD proposes two methods for adaptive retraining: iteration-dependent and data-dependent. In the following, we explain the details of each approach.

AdaptHD Iteration-Dependent Learning: The iteration dependent approach increases the learning rate (α) adaptively during the retraining iterating. AdaptHD starts the training with a large learning rate, α_{max}, in order to speedup the training. Then, it reduces the learning rate depending on the changes in the classification accuracy. AdaptHD exploits the error rates (percentages of wrong classifications) of the last β iterations in order to decide the learning rate on the next iteration. Where there is larger the error rates on the last β iterations, AdaptHD selects a larger learning rate in order to faster adjust the model. As the changes in the error rate decreases, the accuracy gets closer to converging, so AdaptHD uses lower learning rates.

Figure 3 shows the overview of the AdaptHD using iteration-dependent adaptive learning framework. After initial training, the HD model is retrained iteratively on every training data points. If the HD model misclassifies a training data, the HD model is updated by adding and subtracting αQ from two hypervectors in the model. Once the HD model has been retrained on the entire training set, the previous error rates are sent to the Learning Rate Decider *LRDecider* to set α for the next iteration. Figure 3 also shows how the *LRDecider* sets the α value on the next iterations. Based on the data distribution of error rates, AdaptHD uses a linear function to determine α value. To avoid floating point operations, we use a step function to set the learning rate depending on the average error rate. As the error rate decreases, α also decreases. This is because as the accuracy of our HD model gets closer

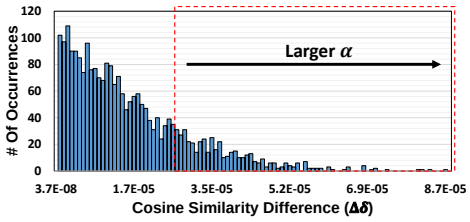

Fig. 4. Cosine difference ($\Delta\delta$) distribution in data-dependent.

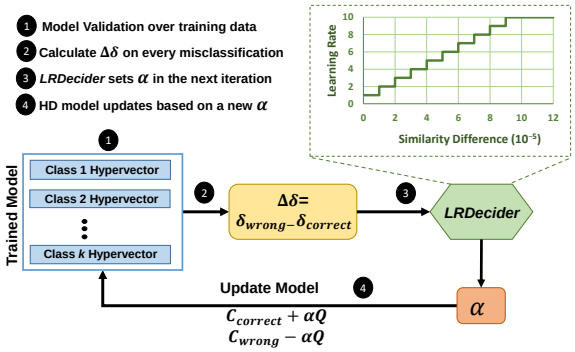

Fig. 5. AdaptHD framework supporting data-dependent training.

to converging, the error rate will decrease. Thus, α should be decreased to help the model converge faster.

AdaptHD Data-Dependent Learning: Unlike the iteration depended approach which uses the same learning rate for all data points, data-dependent approach adaptively modifies the learning rate for each sample data. The goal of this approach is to make higher changes to the model if a data point is misclassified with a far distance. Figure 4 shows the histogram of the cosine similarity difference in speech recognition application [10] during a single retraining iteration. Our evaluation shows that most of the data points have small cosine similarity difference. Meanwhile, there are some outliers (highlighted in the figure) that have much larger similarities differences. Since the model is further off from these training hypervectors, the HD model should be adjusted more aggressively to address that. To this end, AdaptHD uses a large learning rate for data points which are misclassified with the HD model with a far distance, while using smaller learning rates for marginal misclassifications. AdaptHD uses the difference of the cosine similarity of a query with the correct class, $C_{correct}$, and misclassified class, C_{wrong}, as a similarity metric to identify how far a misclassification occurs. When the difference between the cosine similarities is larger, we need to adjust both class hypervectors more to achieve this goal. When the difference is smaller, the model adjustment can happen with smaller learning rate.

Figure 5 shows the overview of AdaptHD framework using data-dependent learning. First, AdaptHD checks the similarity of each training data point with the current HD model. If a data point is misclassified, the difference of cosine similarities of a correct and incorrect class, $\Delta\delta = \delta(Q, C_{wrong}) - \delta(Q, C_{correct})$, is sent to the *LRDecider*, where $\delta(*)$ denotes cosine similarity. The *LRDecider* uses this similarity difference to decide what α should be for that data point. The HD model is then updated with the data point, Q, multiplied by α. Once the HD model has been updated, AdaptHD sends a next data point to check the quality of the model. Figure 5 shows how the *LRDecider* block determines the learning rate for each data point. As shown, α is determined with a linear function based on the similarity difference. We use a step function to set the learning rate depending on the similarity difference. As the similarity difference increases, α also increases in order to make a larger modification on the current model.

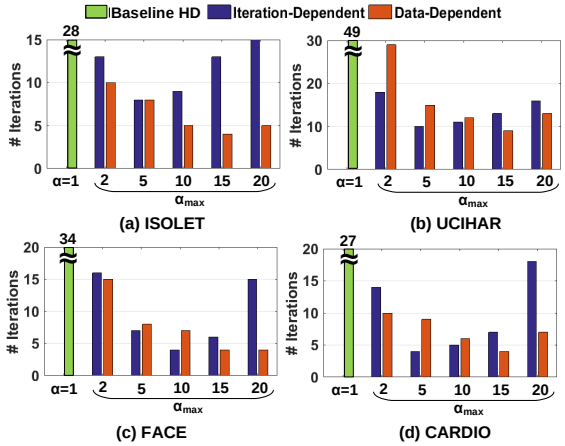

Fig. 6. Number of required training iterations of the baseline HD and AdaptHD using iteration-dependent and data-dependent approaches.

III. Evaluation

A. Experimental Setup

We describe the functionality of the proposed AdaptHD using C++ implementation. We run AdaptHD on an embedded device (Raspberry Pi 3) using ARM Cortex A53 CPU. For the measurement of CPU power, we use a Hioki 3334 power meter. All experiments are performed for a case of using $D = 10,000$ dimensions. We evaluate the efficiency of the proposed AdaptHD on four practical classification problems: Speech Recognition (ISOLET) [10], Activity Recognition (UCIHAR) [11], Face Recognition (FACE) [9], and Cardiotocography (CARDIO) [12].

B. AdaptHD & Maximum Learning Rate

Training Iterations: There is an optimal α_{max} that the *LRDecider* function should saturate at. This is because if the function saturates at too low of an α_{max}, retraining may still take many iterations. In this case, our method is closer to having a low constant learning rate as discussed in Section II-B. However, if α_{max} is large, we may run into the same problem of having a large learning rate. This fluctuates the accuracy of the HD model and result in a low accuracy or divergence.

We explored different values of α_{max} to find the optimal value. Figure 6 shows the effect of increasing α_{max} for iteration-dependent learning. As we predicted, a small α_{max} of 2 does not significantly reduce the number of training iterations. Additionally, AdaptHD using a large α_{max} of 20 takes many iterations to converge because the HD model is overadjusted. In some cases, the HD model diverges because it is overadjusted, such as when α_{max} is 20 for the ISOLET dataset. By looking at Table I, an α_{max} of 5 is the optimal value for the iteration-dependent approach. Figure 6 shows the impact of α_{max} on the number of retraining iterations of data-dependent approach. Our results show that the optimal α_{max} for the data-dependent approach is 15, which is slightly higher than the optimal value of the iteration-dependent approach. In fact, the data-dependent approach needs to make higher modification on each individual data, while still keeping the learning rate of marginally misclassified data small. However, iteration-dependent needs to be more conservative since it uses the same learning rate for all data points.

AdaptHD Accuracy: Table I shows the classification accuracy of AdaptHD using different α_{max}. The results are reported for four different cases: AdaptHD with iteration-based and data-dependent learning rates, AdaptHD using fixed learning rate of α_{max}, and the baseline HD ($\alpha = 1$). In terms of accuracy, AdaptHD provides higher accuracy using small α_{max}, e.g., 5 or 10. Using a very large learning rate, the fluctuation in the accuracy eliminates AdaptHD accuracy to converge into a stable value. Our evaluation shows

TABLE I
THE IMPACT OF THE MAXIMUM LEARNING RATE (α_{max}) ON THE ADAPTHD CLASSIFICATION ACCURACY.

Learning Rate	$\alpha_{max}=2$			$\alpha_{max}=5$			$\alpha_{max}=10$			$\alpha_{max}=15$			$\alpha_{max}=20$			$\alpha=1$
	Iteration	*Data*	α_{max}	*Iteration*	*Data*	α_{max}	*Iteration*	*Data*	α_{max}	*Iteration*	*Data*	α_{max}	*Iteration*	*Data*	α_{max}	
ISOLET	91.28	91.34	91.66	91.34	91.89	90.21	91.85	90.83	89.05	92.24	91.15	88.17	90.89	90.13	N/A	91.10
UCIHAR	93.44	94.63	93.96	96.20	94.40	93.9	95.62	94.41	93.65	94.66	94.34	92.12	94.34	92.66	NA	93.82
FACE	94.35	94.69	94.91	95.97	95.83	95.81	95.75	95.87	95.03	95.15	95.03	94.11	94.23	95.71	93.2	94.38
CARDIO	99.88	98.34	98.56	99.90	99.53	99.30	100	98.59	99.21	100	98.12	98.12	99.06	98.59	97.84	98.17

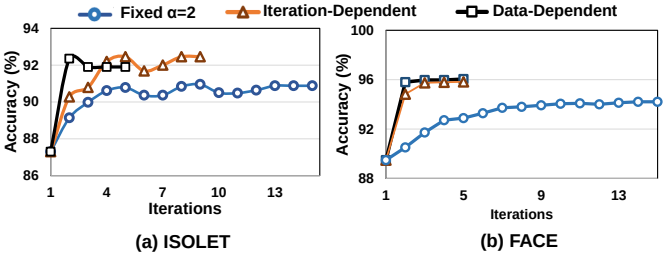

Fig. 7. Classification accuracy of the baseline HD and AdaptHD using iteration and data-dependent approaches.

TABLE II
ACCURACY AND TRAINING EFFICIENCY OF ADAPTHD USING SIMULTANEOUS USE OF ITERATION AND DATA-DEPENDENT APPROACH.

		ISOLET	UCIHAR	FACE	CARDIO
Iteration Dependent	*Energy Improv.*	2.83×	3.93×	5.20×	4.92×
	Speedup	3.08×	4.41×	5.71×	5.29×
Data Dependent	*Energy Improv.*	6.28×	4.90×	7.63×	5.96×
	Speedup	6.83×	5.33×	8.29×	6.51×
Hybrid	*Energy Improv.*	6.25×	5.61×	7.58×	5.94×
	Speedup	6.78×	6.09×	8.24×	6.48×
	Accuracy	92.4%	96.0%	95.9%	99.9%

that AdaptHD trained with iteration-dependent and data-dependent approaches can achieve on average 2.0% and 1.2% higher classification accuracy as compared to the baseline HD (with a fixed α of 1) in a significantly lower number of retraining iterations. Figure 7 visualizes the classification accuracy of AdaptHD using iterating-dependent and data-dependent approaches for ISOLET and FACE applications. Our evaluation shows that HD computing using a fixed and small learning rate takes a very long time to converge. However, AdaptHD exploits a large learning rate to accelerate the first training iterations and a small learning rate for convergence.

C. Hybrid AdaptHD

Table II shows AdaptHD efficiency and accuracy exploiting both iteration and data-dependent approaches. Before each retraining iteration, AdaptHD first sets the learning rate using the iteration-dependent approach. Then, it exploits the data-dependent approach to find a learning rate for each individual data point in an iteration. AdaptHD selects the average learning rate of the iteration and data-dependent approach as a learning rate for each data point. The results (on ARM Cortex A53 CPU) show that AdaptHD in hybrid mode can achieve the high classification accuracy of iteration-dependent and the fast training of data-dependent approach. For example, AdaptHD using iteration-dependent (data-dependent) approach can achieve 4.6× and 4.2× (6.7× and 6.2×) speedup and energy efficiency improvement during training as compared to the baseline HD computing algorithm.

D. AdaptHD Comparison with Other Classifiers

Finally, we compare the efficiency and accuracy of AdaptHD with other light-weight classification algorithms including SVM, Random Forest, Naive Bayes, Gradient Boosting (GBoosting), and Perceptron. We exploited Scikit-learn library [13] for the model training and testing and used grid search to find the best hyperparameters. Table III lists the classification accuracy and the training/test execution time of all algorithms running on an embedded device (Raspberry Pi 3) using ARM Cortex A53 CPU. Our evaluation shows that AdaptHD can provide better or comparable accuracy to other algorithms. For example, thanks to our non-linear encoding, AdaptHD can provide about 4-5% higher accuracy than

TABLE III
ACCURACY, TRAINING AND TESTING EXECUTION TIME OF ADAPTHD WITH OTHER LIGHT-WEIGHT ALGORITHMS.

	Training Execution (s) ISOLET	UCIHAR	FACE	CARDIO	Accuracy	Inference Execution (ms)
SVM	4.82	2.81	0.53	0.03	95.69%	6.91
Random Forest	1.04	1.10	2.96	0.02	93.50%	5.66
Naive Bayes	0.21	0.78	0.50	0.06	87.92%	1.85
GBoosting	53.15	7.36	77.41	0.21	96.05%	4.38
Perceptron	1.05	1.89	1.21	0.06	91.78%	0.80
Iteration (AdaptHD)	1.36	1.40	0.73	0.01	96.15%	
Data (AdaptHD)	0.68	1.26	0.41	0.009	95.38%	1.24
Hybrid (AdaptHD)	0.66	1.16	0.40	0.008	96.05%	

perceptron algorithm which performs iterative training on original data. In terms of efficiency, AdaptHD can provide much faster computation in both training and testing. This higher AdaptHD efficiency comes from the following points: (i) AdaptHD starts the retraining process from an initial model which significantly reduces the number of required iterations. (ii) AdaptHD represents data points using a binary vectors which can process with significantly higher efficiency as compared to non-binary vectors. (iii) AdaptHD simplifies the inference task to Hamming distance similarity check, which can be implemented with an order of magnitude higher efficiency than dot product. Our evaluation shows that AdaptHD can achieve 2.1× faster training and 5.5× testing as compared to learning algorithms such as SVM.

IV. CONCLUSION

In this paper, we propose two adaptive HD retraining approaches which enable retraining the HD model to be more efficient while maintaining the same accuracy. AdaptHD changes the learning rate to converge the accuracy faster. The iteration-dependent changes the learning rate through the retraining iterations, while data-dependent approach adaptively changes learning rate depending on each data. Our evaluation shows that AdaptHD can achieve 6.9× speedup and 6.3× energy efficiency improvement during training as compared to the state-of-the-art HD computing algorithm.

V. ACKNOWLEDGEMENTS

This work was partially supported by CRISP, one of six centers in JUMP, an SRC program sponsored by DARPA, and also NSF grants #1911095, #1826967, #1730158 and #1527034.

REFERENCES

[1] 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.
[2] P. Kanerva, "What we mean when we say "whats the dollar of mexico?": Prototypes and mapping in concept space," in *AAAI Fall Symposium*, pp. 2–6, 2010.
[3] A. Joshi *et al.*, "Language geometry using random indexing," *Quantum Interaction 2016 Conference Proceedings*, In press.
[4] A. Rahimi *et al.*, "A robust and energy-efficient classifier using brain-inspired hyperdimensional computing," in *ACM ISLPED*, pp. 64–69, ACM, 2016.
[5] O. Räsänen *et al.*, "Modeling dependencies in multiple parallel data streams with hyperdimensional computing," *IEEE Signal Processing Letters*, vol. 21, no. 7, pp. 899–903, 2014.
[6] O. Rasanen *et al.*, "Sequence prediction with sparse distributed hyperdimensional coding applied to the analysis of mobile phone use patterns," *IEEE TNNLS*, vol. PP, no. 99, pp. 1–12, 2015.
[7] M. Imani *et al.*, "Hierarchical hyperdimensional computing for energy efficient classification," in *DAC*, p. 108, ACM, 2018.
[8] A. Rahimi *et al.*, "Hyperdimensional computing for noninvasive brain–computer interfaces: Blind and one-shot classification of eeg error-related potentials," in *BICT*, 2017.
[9] G. Griffin *et al.*, "Caltech-256 object category dataset," 2007.
[10] "Uci machine learning repository." http://archive.ics.uci.edu/ml/datasets/ISOLET.
[11] "Uci machine learning repository." https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones.
[12] "Uci machine learning repository." https://archive.ics.uci.edu/ml/datasets/cardiotocography.
[13] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2825–2830, 2011.