PQ-HDC: Projection-Based Quantization Scheme for Flexible and Efficient Hyperdimensional Computing

Chi-Tse Huang, Cheng-Yang Chang, Yu-Chuan Chuang, An-Yeu (andy) Wu

PQ-HDC: Projection-based Quantization Scheme for Flexible and Efficient Hyperdimensional Computing

Chi-Tse Huang(✉), Cheng-Yang Chang, Yu-Chuan Chuang, and An-Yeu (Andy) Wu

Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan
{ rickhuang, kevin, frankchuang }@access.ee.ntu.edu.tw, andywu@ntu.edu.tw

Abstract. Brain-inspired Hyperdimensional (HD) computing is an emerging technique for low-power/energy designs in many machine learning tasks. Recent works further exploit the low-cost quantized (bipolarized or ternarized) HD model and report dramatic improvements in energy efficiency. However, the quantization loss of HD models leads to a severe drop in classification accuracy. This paper proposes a projection-based quantization framework for HD computing (PQ-HDC) to achieve a flexible and efficient trade-off between accuracy and efficiency. While previous works exploit thresholding-quantization schemes, the proposed PQ-HDC progressively reduces quantization loss using a linear combination of bipolarized HD models. Furthermore, PQ-HDC allows quantization with flexible bit-width while preserving the computational efficiency of the Hamming distance computation. Experimental results on the benchmark dataset demonstrate that PQ-HDC achieves a 2.82% improvement in accuracy over the state-of-the-art method.

Keywords: Brain-inspired computing, hyperdimensional computing, dynamic model, energy efficiency

1 Introduction

In the era of the Internet of Things (IoT), a huge amount of data is generated daily. Machine learning (ML) models on edge devices spring up to collect and analyze these data. Nowadays, deep neural network (DNN) is the most prevalent ML models. However, DNN requires considerable computational resources with iterative gradient decent and a huge number of parameters, which are not available on resource-constrained edge devices. Therefore, a lightweight alternative is more desirable for efficient model deployment. Brain-inspired Hyperdimensional Computing (HDC) [1] has been proposed as a lightweight classifier with the low training cost, which emulates the patterns of neural activities by computing with

high-dimensional (HD) vectors, called hypervectors (HVs). HDC exhibits many remarkable characteristics, including high energy efficiency [2] and few-shot learning ability [3]. Moreover, HDC achieves the great success in various real-world applications, such as text classification [4], speech recognition [5], genome sequencing [6], and emotion recognition [7]. These properties make HDC suitable for efficient deployment on edge devices and realistic applications.

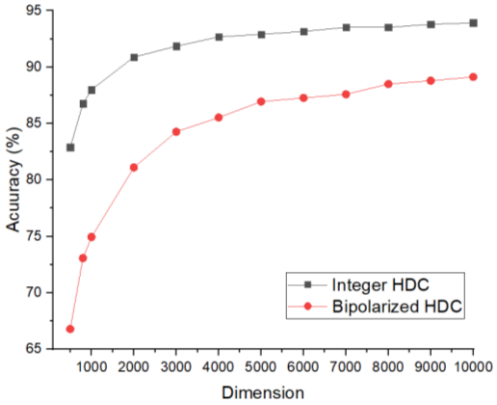

Fig. 1 The accuracy of the integer and bipolarized HD models.

However, to provide acceptable accuracy on realistic classification problems, most works calculate computation-intensive cosine similarity between HVs with integer representation. On the contrary, the quantized HD model trades classification accuracy for drastically improved computational efficiency. As the work of [8] pointed out, the integer HD model consumes much more energy than the bipolarized one. To illustrate this idea, Fig. 1 compares the classification accuracy of full-precision (integer) and quantized (bipolarized) HD models with different HD dimensions. Fig. 1 shows that the bipolarized HD model suffers from a significant accuracy drop. While QuantHD [9] introduces a retraining mechanism which iteratively adjusts the model on data with wrong predictions to close the accuracy gap, there are still some open issues:

1) There is still a large gap between the classification accuracy of the bipolarized HD model and that of the integer one. Therefore, quantizing HD models with less information loss is worth investigating.
2) The ultra-low complexity of the Hamming distance computation is only applicable to one-bit bipolarized HD models. For HD models quantized to multiple bits, computation-intensive cosine similarity is required. Thus, the lack of flexibility in bit-width for representing HD models should be addressed.

In this paper, we propose a Projection-based Quantization framework for HDC (PQ-HDC) to improve the flexibility and the accuracy of quantized HD models. QuantHD [9] utilizes a single bit to quantize the integer HVs into bipolarized ones,

leading to large quantization loss. Instead, PQ-HDC uses multiple bits to shrink the quantization loss progressively while preserving the computational efficiency of the Hamming distance computation. The main contributions of this paper are summarized as follows:

1) **Multi-bit quantized HD model using Hamming distance metric:** PQ-HDC acts as an intermediate between the integer and bipolarized HD models. Compared to traditional methods which quantize the integer HD model into a single bipolarized one, PQ-HDC utilizes a linear combination of bipolarized AMs to reduce the quantization loss progressively. Additionally, by normalizing the combination of bipolarized HVs in each class, PQ-HDC can quantize HD models with arbitrary bit-width while preserving the computational efficiency of the Hamming distance computation.
2) **Improve the computation-efficient:** Compared to the state-of-the-art [9] quantized HDC scheme based on the Hamming distance computation, PQ-HDC improves 2.82% and 0.79% in accuracy on MNIST and ISOLET datasets, respectively. To show that PQ-HDC achieves a scalable accuracy–complexity trade-off, we also discuss the memory and the computational complexity at the end of this article.

2 Review of Hyperdimensional Computing

HDC is based on the high-dimensional and dense bipolarized HVs. In the training phase, HDC transforms training data into bipolarized HVs and aggregates those bipolarized HVs of the same class to construct the class HVs. In the inference phase, testing data forms query HVs through the same transformation procedures. Then, HDC checks similarities between the query HV and the class HVs. Finally, the class with the highest similarity is the prediction.

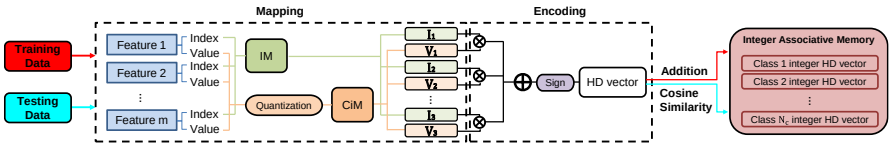

Fig. 2 The processing flow of HDC.

Mapping

In HDC, the goal of mapping is to project the information of a feature vector $X = [x_1, x_2, \dots, x_m]^T$ with m features into D-dimensional HD space. Each feature identifier (ID) and its value are mapped to a pair of bipolarized HVs through an Item Memory (IM) and a Continuous Item Memory (CiM), respectively. IM identifies the feature information of every element in the feature vector X. IM =

$\{I_1, I_2, \dots, I_m\}, where\ I_i \in \{-1, +1\}^D\ for\ i = 1 \sim m.$ I_i is the item HV that represents the ID of x_i in X. To isolate the feature information, the bipolarized HVs in IM are approximately orthogonal with each other. In other words, $\frac{Hamming(I_i, I_j)}{D} \cong 0.5, i \neq j,$ where $Hamming(\cdot,\cdot)$ denotes the Hamming distance between two HVs. To achieve this purpose, HDC assigns $\{I_1, I_2, \dots, I_m\}$ as m independently and randomly generated bipolarized HVs.

Meanwhile, CiM preserves the magnitude information of every element in the feature vector X. CiM quantizes the numerical range of feature values into l levels, each of which corresponds to a quantization level. In other words, CiM = $\{CI_1, CI_2, \dots, CI_l\}$ and $\frac{Hamming(CI_i, CI_j)}{D} = \frac{||i-j||}{l-1}, i \neq j$. This design maintains the spatial relationship of quantization levels. To achieve this purpose, HDC assigns CI_1 as one randomly generated bipolarized HV. Then, HDC flips $\frac{D}{l-1}$ bits which were not flipped to generate the continuous item HV of the next level. The same bit-flipping mechanism applies to other HVs in CiM.

Based on IM and CiM, HDC maps the feature vector X into the HD space. Specifically, x_i is mapped to a pair of HVs, consisting of the HV from IM I_i and the HV from CiM V_i. Note that V_i is selected from $\{CI_1, CI_2, \dots, CI_l\}$ according to the actual value of x_i. After the HDC mapping stage, the feature vector X will be mapped to m pairs of bipolarized HVs (I_i, V_i).

Encoding

The encoding stage of HDC combines m pairs of HVs into one bipolarized HV that represents the original feature vector X. Specifically, for each pair of HVs, HDC binds the feature information and magnitude information together by multiplying I_i with V_i. $I_i * V_i$ results in a single HV that represents x_i in the feature vector X.

Next, HDC bundles these resulting encoded HVs (i.e. $I_i * V_i$) by summarizing them into an integer HV. This integer HV $T^{int} \in R^D$ is the representation of X in the HD space. Finally, HDC bipolarizes this integer HV T^{int} into bipolarized HV T^{bip} by the $sign$ function.

$$T^{bip} = sign[T^{int}] = sign[\, I_1 * V_1 + I_2 * V_2 + \cdots + I_m * V_m \,]. \qquad (1)$$

Training

In the training phase, all training data go through the same mapping and encoding stages mentioned above. Then, the resulting HVs are sent to the associative memory (AM) for aggregation. In detail, the set of HVs that correspond to training data belonging to the same class are summed up to form a class HV. For a N_c-class

classification task, HDC stores N_c class HVs in the AM, as shown in Fig 2. For the j^{th} class, the corresponding integer class HV (AM_j^{int}) is computed as:

$$AM_j^{int} = \sum T^{bip}, T^{bip} \; which \; belongs \; to \; class \; j, \quad and \; j \leq N_c(number \; of \; class). \tag{2}$$

Besides, HDC often bipolarizes AM^{int}, resulting in AM^{bip} for both storage-efficiency and computational-efficiency. Bipolarized AM only consumes a single bit to represent each element in a class HV. Moreover, bipolarization reduces inference computation from cosine similarity to Hamming distance.

Inference

In the inference phase, the testing data is transformed to a bipolarized query HV $Q \in \{-1, 1\}^D$ by the same processing flow of mapping and encoding in the training phase. As shown in Fig. 2, since AM stores the most representative class HV of each class, HDC makes predictions by checking similarities (S^{int}) between Q and all integer class HVs. Then, the class with the highest similarity is the prediction of this data. On the other hand, bipolarized HD model has significantly lower computational complexity and energy cost [10]; however, bipolarization leads to a drop in the accuracy due to the quantization loss.

$$prediction^{int} = \underset{class}{\operatorname{argmax}} S_{class}^{int}. \tag{3}$$

3 Proposed Projection-based Quantization for Hyperdimensional Computing (PQ-HDC)

PQ-HDC quantizes the integer class HV AM_j^{int} into multiple bipolarized class HVs and weights. Specifically, the linear combination of these bipolarized class HVs approximates the values in the integer class HV. In the training phase, PQ-HDC sequentially finds N_p, the number of bipolarized class HVs used to recover a single integer class HV, bipolarized class HVs to progressively reduce the quantization loss. Note that N_p also denotes the bit-width of the quantized HDC model. The weights for each bipolarized class HV are derived through linear regression and normalization techniques. In the inference phase, PQ-HDC checks the pseudo-hamming distances (PHD), which is defined as the weighted Hamming distance between the query HV and each set of bipolarized class HVs. The class with the lowest PHD value is the prediction.

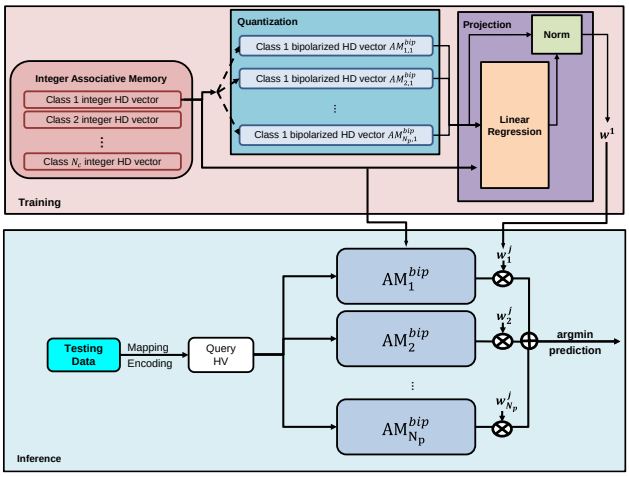

Fig. 3. System flow of the proposed PQ-HDC.

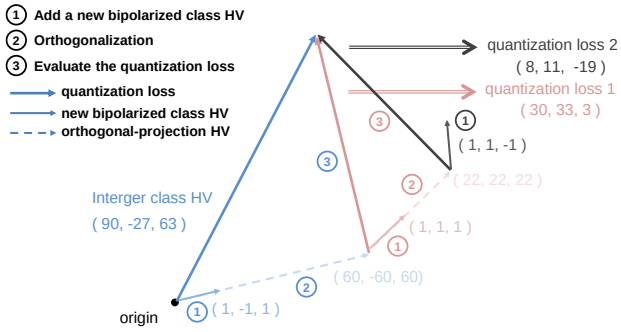

Fig. 4. Procedure in the training phase of PQ-HDC that generates a bipolarized class HV and updates HV^{loss} for the next iteration.

3.1 Off-line Training Phase

Given an integer AM, PQ-HDC transforms the integer class HV of each class into N_p bipolarized class HVs. The procedure of PQ-HDC off-line training phase is summarized in Algorithm 1 and depicted in Fig.4, respectively. Since each class is processed independently and separately, we take the j^{th} class as an example in the following description.

First, PQ-HDC defines HV_j^{loss}, which is an HV that records the quantization loss in each HD dimension, as the target that the bipolarized HVs aim to approximate. The initialization procedure of PQ-HDC includes setting HV_j^{loss} as AM_j^{int}. Next, as shown in Fig. 4, we define a procedure that contains three steps. PQ-HDC iterates over this procedure N_p times, each of which generates a pair of HVs, denoted as

$AM_{i,j}^{bip}$, and quantization loss HV (HV_j^{loss})for the next iteration. The three steps in the procedure are described as follows:

Step 1. Add a new bipolarized class HV: PQ-HDC acquires $AM_{i,j}^{bip}$, which represents the bipolarized class HV of the j^{th} class in the i^{th} iteration, by bipolarizing HV_j^{loss}. This step is depicted in row 5 of Algorithm 1.

$$AM_{i,j}^{bip} = sign\left[HV_j^{loss} \right]. \tag{4}$$

Step 2. Orthogonalization: Since $AM_{i,j}^{bip}$ and HV_j^{loss} are not parallel to each other, PQ-HDC decomposes HV_j^{loss} into two components. The first component is parallel to $AM_{i,j}^{bip}$ while the second one is orthogonal to $AM_{i,j}^{bip}$. Note that the decomposition can be simply done by orthogonal projection. Specifically, PQ-HDC calculates the orthogonal-projection HV (HV_j^{op}) of HV_j^{loss} on $AM_{i,j}^{bip}$. This step is depicted in row 6 of Algorithm 1. The resulting HV_j^{op} is the parallel component of HV_j^{loss} on $AM_{i,j}^{bip}$.

$$HV_j^{op} = AM_{i,j}^{bip} \left[\left(HV_j^{loss} \cdot AM_{i,j}^{bip} \right) / \left| \left| AM_{i,j}^{bip} \right| \right|^2 \right]. \tag{5}$$

Step 3. Evaluate the quantization loss: To ensure that the updated HV_j^{loss} for the next iteration is orthogonal to $AM_{i,j}^{bip}$, PQ-HDC subtracts HV_j^{loss} with HV_j^{op}. This step is depicted in row 7 of Algorithm 1.

After iterating the above procedure N_p times, a set of bipolarized class HVs $S = \{AM_{i,j}^{bip} for\ i = 1,2,\dots,N_p\}$ is generated. Note that the HVs in S are not mutually orthogonal. To approximate the integer class HV AM_j^{int} based on the linear combination of HVs in S, PQ-HDC uses linear regression to project AM_j^{int} onto the span of S. Therefore, the weights for the linear combination of S, denoted $\{w_i^j\ for\ i = 1,2,\dots,N_p\}$, minimize the quantization loss, as shown in equation (6). This step is depicted in rows 8-11 of Algorithm 1.

$$Recover\left(AM_j^{int} \right) = \sum_{i=1}^{N_p} w_i^j\, AM_{i,j}^{bip}. \tag{6}$$

$$Norm\left(Recover\left(AM_{c1}^{int} \right) \right) \neq Norm\left(Recover\left(AM_{c2}^{int} \right) \right), if\ c1 \neq c2. \tag{7}$$

However, the sets of weights for different classes vary. Consequently, the approximated integer class HVs have different norms. This phenomenon is in favor of the class HV with the larger norm for prediction. To unify the norms, PQ-HDC normalizes the weights, depicted in rows 12 in Algorithm 1. Based on the normalization technique, PQ-HDC can perform Hamming distance computation

without considering the norm of each class HV

Algorithm 1 Off-line Training Phase of PQ-HDC

Input: AM^{int} – integer AM with N_c classes
N_p – number of PQ AMs

Output: $AM^{bip} = \{AM_1^{bip}, AM_2^{bip}, \dots, AM_{Np}^{bip}\}$ – PQ AMs
$w = \{w^1, w^2, \dots, w^{N_c}\}$ – weights of classes

1: **for** ($class$ in 1: N_c) **do**
2: $X \leftarrow R^{[N_p, D]}$
3: $HV_{class}^{loss} \leftarrow AM_{class}^{int}$
4: **for** ($index$ in 1: N_p) **do**
5: $AM_{index,class}^{bip} = sign[\, HV_{class}^{loss}]$
6: $HV_{class}^{op} \leftarrow AM_{index,class}^{bip}[HV_{class}^{loss} \cdot AM_{index,class}^{bip} / \left|\left|AM_{index,class}^{bip}\right|\right|^2]$
7: $HV_{class}^{loss} \leftarrow HV_{class}^{loss} - HV_{class}^{op}$
8: $X[index\text{-}1] \leftarrow AM_{index,class}^{bip}$
9: $Y \leftarrow AM_{class}^{int}$
10: $P \leftarrow X^T(XX^T)^{-1}$
11: $w^{class} \leftarrow YP$
12: $w^{class} \leftarrow \sqrt{D}\, w^{class} / ||w^{class} * AM_{class}^{bip}||$

3.2 On-line Inference Phase

The Inference flow of PQ-HDC is depicted in Fig. 3. In the inference phase, testing data undergoes the same processing flow in the training phase to become a query HV Q, as shown in equation (1).

Based on the normalization, PQ-HDC can reduce the cosine similarity between Q and the recovering integer class HV to the weighted Hamming distance PHD between Q and the set of bipolarized class HVs like equation 8, where $COS(\cdot,\cdot)$ denotes the cosine similarity between two HVs.

$$S_j^{int} \cong \widetilde{S_j^{int}} = COS\left((Q, Recover\left(AM_j^{int}\right)\right) = 1 - 2\frac{PHD\left(Q, Recover\left(AM_j^{int}\right)\right)}{D}. \quad (8)$$

$$PHD\left(Q, Recover\left(AM_j^{int}\right)\right) = \frac{D}{2} + \sum_{i=1}^{N_p} w_i^j \left(Hamming\left(Q, AM_{i,j}^{bip}\right) - \frac{D}{2}\right). \quad (9)$$

Taking the class j as an example, PQ-HDC evaluates the Hamming distance

between Q and all bipolarized class HVs of the class j separately. Namely, $H_j = \{Hamming(Q, AM_{i,j}^{bip}) for\ i = 1,2,\dots,N_p\}$ is generated. Then, PQ-HDC multiplies these Hamming distances in H_j with their corresponding weights to calculate the PHD between Q and the recovering integer class HV by (9). The same PHD computation mechanism applies to other classes. Finally, the class with the lowest PHD value is the prediction. The algorithm of the PQ-HDC on-line inference phase is summarized in Algorithm 2.

Algorithm 2 On-line Inference Phase of PQ-HDC

Input: D_{test} – testing data
N_p – number of PQ AMs
$AM^{bip} = \{AM_1^{bip}, AM_2^{bip}, \dots, AM_{Np}^{bip}\}$ – PQ AMs
$w = \{w^1, w^2, \dots, w^{N_c}\}$ – weights of classes

Output: Prediction Pre_{test}

Transform D_{test} to Q by (1)
$PHD \leftarrow R^{[N_c]}$
for ($class$ in 1: N_c) **do**
  for (i in 1: N_p) **do**
    $H_{class,i} = Hamming(Q, AM_{i,class}^{bip})$
  $PHD_{class} = \frac{D}{2} + \sum_{k=1}^{N_p} \left(w_k^{class} \left(H_{class,k} - \frac{D}{2} \right) \right)$
$Pre_{test} = \arg\min_{class}(PHD_{class})$

4 Experimental Settings and Simulation Results

4.1 Dataset and Simulation Settings

We conducted experiments on MNIST [11] and ISOLET [12] datasets, which are both common benchmarks for HDC. Our proposed PQ-HDC is implemented in Python running on AMD Core TR-2950X processor with 128 GB memory. To evaluate the effectiveness of PQ-HDC, we compare our framework with HD models with full-precision (integer) and quantized representations based on QuantHD [9]. Classification accuracy, storage requirement, and computational complexity of each method are summarized in Table 1. All experiments are conducted over 10 independent runs to obtain the averaged simulation results.

Table 1. Summary of storage and computational complexity between different models

Model	Averaged Accuracy	Normalized Storage	Computational Complexity
Integer HD model	93.36%	1	O($D * N_c$) times multiplications + O($D * N_c$) times additions
Bipolarized HD model [9]	90.15%	0.03125	O($D * N_c$) times additions
Ternarized HD model [9]	91.68%	0.0625	O($D * N_c$) times multiplications + O($D * N_c$) times additions
PQ2 HD model ($N_p = 2$)	**91.95%**	**0.0627**	**O($N_p * N_c$) times multiplications + O($N_p * D * N_c$) times additions**
PQ3 HD model ($N_p = 3$)	92.75%	0.09405	O($N_p * N_c$) times multiplications + O($N_p * D * N_c$) times additions

4.2 Comparisons of Classification Accuracy

Fig. 5 shows the averaged retrained accuracy of the HD models with different quantization schemes and HD dimensions. The integer HD model has the highest accuracy among all models; however, its memory and computational overhead are not affordable on resource-constrained edge devices. On the contrary, bipolarized HD shows the highest computational efficiency, since it does not require any multiplication operation. Nevertheless, the drop in the accuracy is not acceptable.

As intermediate solutions, the ternarized HDC model can close the accuracy gap between the integer and bipolarized HD models with computation-intensive cosine similarity. Meanwhile, as shown in Fig. 5, the PQ2($\boldsymbol{N_p = 2}$) HDC model can improve 2.82% and 0.79% in accuracy against the bipolarized HD model on the MNIST and ISOLET dataset, respectively ($\boldsymbol{D}$ = 10000). Note that the storage overhead of PQ2 HDC model is comparable to the ternarized HD model.

4.3 Performance Analysis

For PQ-HDC, since the extra storage space for weights (32$*$ $\boldsymbol{N_p}$ bits for a class) is negligible compared to high-dimension HVs, the storage complexity is O($\boldsymbol{N_p} * \boldsymbol{D}$). Namely, the storage complexity of the PQ2 HDC model is equal to that of the ternarized HD model with higher classification accuracy.

Next, we evaluate the computational complexity for inferencing on testing data in terms of operation counts. The inference procedures of integer and ternarized HD models include computing $\boldsymbol{N_c}$ times cosine similarities, where O($\boldsymbol{D}$) times multiplications and O($\boldsymbol{D}$) times additions are required. Meanwhile, bipolarized HD model computes $\boldsymbol{N_c}$ Hamming distance, where only O($\boldsymbol{D}$) times additions are

required. Finally, the PHD computation of PQ-HDC includes the calculation of $\boldsymbol{N_p \times N_c}$ times Hamming distances along with $\boldsymbol{N_p \times N_c}$ times multiplications. Therefore, PQ-HDC is more computational-efficient than the ternarized HD model of the state-of-the-art.

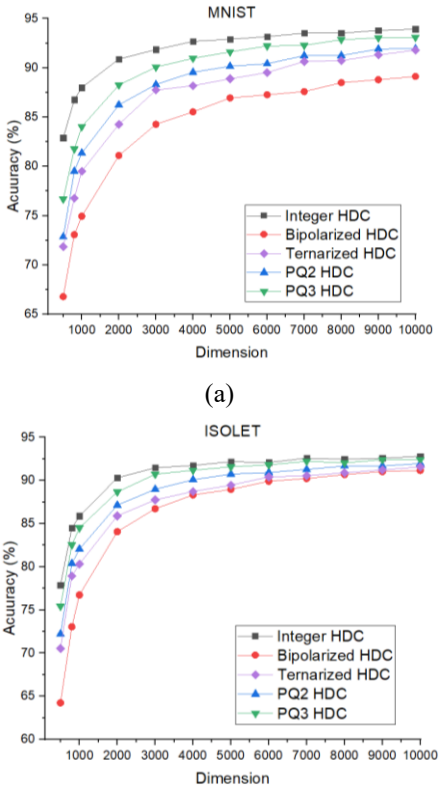

Fig. 5 Comparisons of the averaged accuracy among different frameworks. The line PQi HDC denotes PQ-HDC model with N_p set as i.

5 Conclusions

In this paper, we propose a Projection-based Quantization HD computing (PQ-HDC) framework to effectively improve the accuracy-complexity trade-off. PQ-HDC also shows scalability by utilizing a linear combination of bipolarized AMs to achieve quantization with arbitrary bit-width while preserving the computational efficiency of the Hamming distance computation. The experimental results

demonstrate that PQ-HDC with N_p set as 2 and 3 can save 93.7% and 90.6% memory overhead, respectively, compared to the HDC model with full precision. Furthermore, PQ-HDC also has higher accuracy against ternary HDC with the same storage and lower computation complexity.

Acknowledgments. This work was supported by the Ministry of Science and Technology of Taiwan under Grant MOST 109-2221-E-002-175.

References

1. P. Kanerva, "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors," *Cogn. Comput.*, vol. 1, no. 2, pp. 139–159, 2009.
2. A. Rahimi *et al.*, "Efficient biosignal processing using hyperdimensional computing: Network templates for combined learning and classification of ExG signals," *Proc. IEEE*, vol. 107, no. 1, pp. 123–143, Jan. 2019.
3. A. Rahimi *et al.*, "Hyperdimensional computing for noninvasive brain–computer interfaces: Blind and one-shot classification of EEG error-related potentials," in *Proc. 10th ACM/EAI Int. Conf. Bio-Inspired Inf. Commun. Technol. (BICT)*, 2017, pp. 19–26.
4. F. R. Najafabadi, A. Rahimi, P. Kanerva, and J. M. Rabaey, "Hyperdimensional computing for text classification," Design, Automation Test in Europe Conference Exhibition (DATE), University Booth
5. M. Imani et al., "Voicehd: Hyperdimensional computing for efficient speech recognition," in ICRC, pp. 1–6, IEEE, 2017.
6. M. Imani et al., "Hdna: Energy-efficient dna sequencing using hyperdimensional computing," in BHI, pp. 271–274, IEEE, 2018.
7. E. Chang et al., "Hyperdimensional Computing-based Multimodality Emotion Recognition with Physiological Signals", Proc. IEEE International Symposium on AI for Circuits and Systems (AICAS), pp. 137-141, March 2019.
8. M. Horowitz, "Computing's energy problem (and what we can do about it)," in IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers (ISSCC), Feb. 2014, pp. 10–14
9. M. Imani et al., "Quanthd: A quantization framework for hyperdimensional computing," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), 2019.
10. Yu-Chuan, Chuang; Cheng-Yang, Chang; An-Yeu Andy, Wu., "Dynamic Hyperdimensional Computing for Improving Accuracy-Energy Efficiency Trade-Offs. ," IEEE Workshop on Signal Processing Systems (SiPS). IEEE, 2020.
11. [LeCun et al., 1998a] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE*, 86(11):2278-2324, November 1998. [online] Available: http://yann.lecun.com/exdb/mnist/
12. "UCI ML repository", [online] Available: http://archive.ics.uci.edu/ml/datasets/ISOLET.
13. S. I. Gallant and T. W. Okaywe, "Representing objects relations and sequences", *Neural Comput.*, vol. 25, no. 8, pp. 2038-2078, 2013.