Hyperdimensional Computing with Learnable Projection for User Adaptation Framework

Yu-Ren Hsiao, Yu-Chuan Chuang, Cheng-Yang Chang, An-Yeu (andy) Wu

Hyperdimensional Computing with Learnable Projection for User Adaptation Framework

Yu-Ren Hsiao, Yu-Chuan Chuang, Cheng-Yang Chang, and An-Yeu (Andy) Wu

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

Abstract. Brain-inspired Hyperdimensional Computing (HDC), a machine learning (ML) model featuring high energy efficiency and fast adaptability, provides a promising solution to many real-world tasks on resource-limited devices. This paper introduces an HDC-based user adaptation framework, which requires efficient fine-tuning of HDC models to boost accuracy. Specifically, we propose two techniques for HDC, including the learnable projection and the fusion mechanism for the Associative Memory (AM). Compared with the user adaptation framework based on the original HDC, our proposed framework shows 4.8% and 3.5% of accuracy improvements on two benchmark datasets, including the ISOLET dataset and the UCIHAR dataset, respectively.

Keywords: Brain-Inspired Computing, Hyperdimensional Computing, User Adaptation.

1 Introduction

With the emergence of Internet of Things (IoT), much data is generated by embedded devices [1]. Many IoT applications collect and analyze those data to train machine learning (ML) models in cloud servers and deploy the trained models back to devices for inference. However, performance of deployed models is sensitive to misaligned data distribution caused by subject difference [2] and time-varying properties [3, 4], as depicted in Fig. 1(a). To compensate for the potential performance degradation, a user adaptation framework is required to boost model performance by dynamically adapting models to user data [5-7]. The authors of [2] proposed a cloud-based model adaptation framework that requires edge clients to upload data for retraining. However, the wirelessly transmitted data poses a threat to user's privacy [8] and introduces extra energy consumption. Therefore, an on-device adaptation framework is preferable [9, 10]. However, an ML model with high energy efficiency and fast adaptability is necessary due to limited resources on embedded devices.

Recently, brain-inspired Hyperdimensional Computing (HDC) is emerging as a lightweight alternative to high-complexity ML models. HDC emulates patterns of neural activities in human brains by projecting data into high-dimensional (HD) vectors, called Hypervectors (HVs) [11]. Through exploiting the mathematical properties of HVs in the HD space, HDC has shown the characteristics of high energy

efficiency and fast adaptability in a wide variety of real-world applications, such as image classification [12, 13], speech recognition [14], and bio-signal processing [15]. These advantages make HDC suitable for on-device user adaptation framework, as depicted in Fig. 1(b). However, we argue that the original HDC algorithm ignores significant correlation between features during projection, leading to suboptimal accuracy.

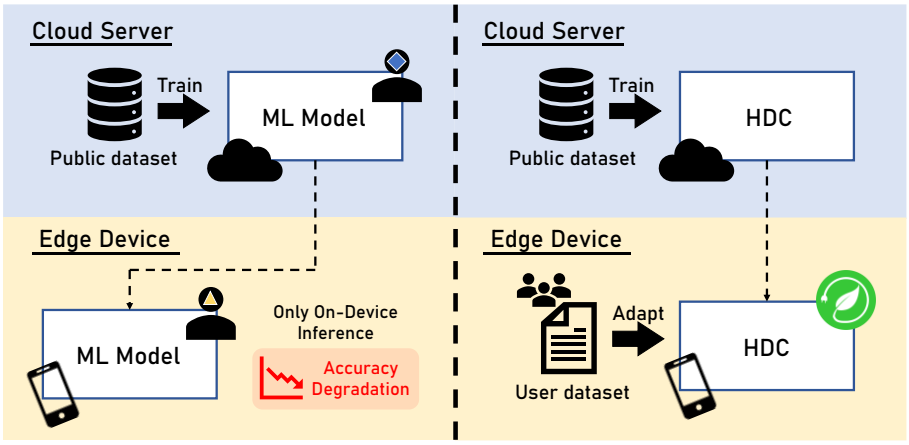


Fig. 1. (a) Without user adaptation, the accuracy of the deployed model easily suffers from misaligned data distribution and time-varying properties. (b) The characteristics of high energy efficiency and fast adaptability make HDC suitable for the user adaptation framework.

To close the knowledge gap, we enable HDC to learn the feature correlation of input data to improve its overall performance. In the cloud, we first transform the original processing flow of HDC into a learnable network, called learnable HDC (L-HDC). L-HDC explicitly emulates the fundamental operations of HDC and learns the feature correlation of input data by backpropagation. After training, we transform L-HDC back to the original HDC, while the weights of L-HDC are kept in the original HDC to perform feature-aware projection. Given the characteristics of high energy efficiency and fast adaptability, HDC models on resource-limited devices can efficiently adapt to the user data. We evaluate the effectiveness of our proposed framework on two benchmark datasets, including the speech recognition dataset ISOLET [16] and the human activity recognition dataset UCIHAR [17]. Based on our simulation results under the settings of a user adaptation framework, our proposed HDC with learnable projection outperforms that with the original projection by 4.8% and 3.5% in ISOLET and UCIHAR, respectively. To the best of our knowledge, this paper first applies HDC with learnable projection to a user adaptation framework to improve classification accuracy.

The rest of the paper is organized as follows. Section 2 describes the algorithm of HDC. Section 3 illustrates our proposed HDC with learnable projection for the user

adaptation framework. Experimental settings and simulation results are shown in Section 4. Finally, we conclude this paper in Section 5.

2 Hyperdimensional Computing

HDC operates with randomly generated bipolar HVs whose components are -1 or 1 with equal probability. The processing flow of HDC is shown in Fig. 2 and includes the following steps.

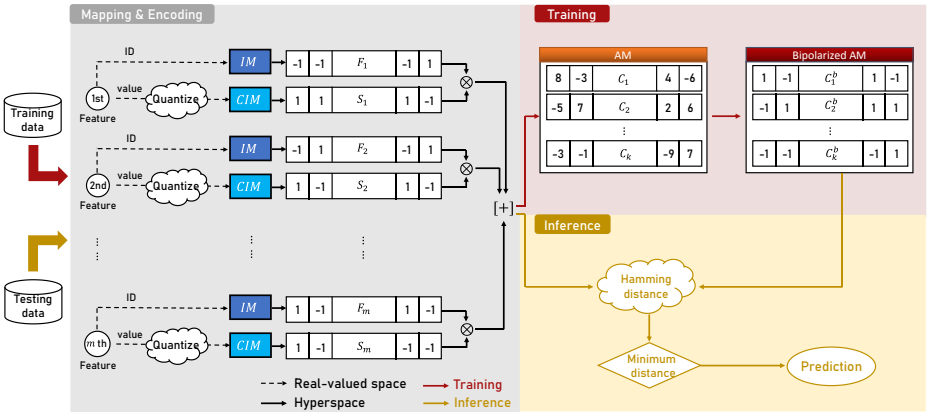


Fig. 2. The processing flow of HDC.

Projection into HD Space. The first step of HDC is to project features into HVs. A feature comprises two parts, i.e., its feature identifier (ID) and its actual value, which are projected into d-dimensional HVs through an Item Memory (IM) and a Continuous Item Memory (CIM), respectively. Assume there are m features in one data sample, $IM = \{F_1, F_2, \dots F_m\} \in \{+1, -1\}^d$, where F_k is the projected HV for the k^{th} feature ID. When d is large enough, every two HVs in IM are nearly orthogonal [18], which means $Hamming(F_i, F_j) \cong 0.5, if\ i \neq j$ and $Hamming(\cdot)$ is the normalized Hamming distance between two vectors. In other words, the projection of the original HDC assumes that there is approximately no correlation among each feature.

CIM is used as a look-up table to project the actual value into an HV. HDC generates the CIM by the following procedures. First, HDC finds the maximum value V_{max} and minimum value V_{min} for each feature. The range between V_{max} and V_{min} is then equally quantized to Q levels, denoted by $\{l_1, l_2, \dots l_Q\} \in \mathbb{Z}$ with l_1 and l_Q corresponding to V_{max} and V_{min}. Each scalar in $\{l_1, l_2, \dots l_Q\}$ is associated with an HV in $CIM = \{L_1, L_2, \dots L_Q\} \in \{+1, -1\}^d$. Namely, L_k is the projected HV for l_k Moreover, to preserve the spatial correlation of neighboring levels, if l_i and l_j are relatively close, L_i and L_j have relatively small Hamming distance. To achieve this, two randomly generated HVs L_1 and L_Q are first assigned to V_{max} and V_{min}, respectively. Then, d/Q randomly selected bits are flipped to generate an HV for the

next level. This process repeats until Q HVs in CIM are generated. The bit-flipping approach ensures a high correlation between adjacent levels, while L_1 and L_Q are orthogonal. For each feature, by looking up the nearest quantized level to its actual value, the HV for its value is selected and denoted as S_k , where $S_k \in CIM, k = 1, 2, \dots, m$. After projecting each feature into HVs, a set of two-vector pairs $P = \{(F_1, S_1), (F_2, S_2), \dots, (F_m, S_m)\}$ is generated for the following step.

Encoding. In this stage, HDC encodes the set P into one representative HV. First, HDC performs the binding operation, an element-wise XOR operation ($\otimes$) between two HVs, to each two-vector pair in the set P and generates m HVs. Then, HDC bundles those m HVs into an encoded HV $T \in \{+1, -1\}^d$ by performing element-wise addition (+) followed by a sign function [·], which is expressed as

$$T = [F_1 \otimes S_1 + F_2 \otimes S_2 + \cdots + F_m \otimes S_m]. \quad (1)$$

Training. After encoding, we denote T_i^j as the encoded HV of the j^{th} data sample corresponding to the i^{th} class. Then, HVs belonging to the i^{th} class are accumulated to form a class HV $C_i \in \mathbb{Z}^d$, which is computed as:

$$C_i = T_i^1 + T_i^2 + \cdots + T_i^{n_i}. \quad (2)$$

n_i is the number of data samples in the i^{th} class. For a k-class classification task, there are k class HVs stored in an Associative Memory $AM = \{C_1, C_2, \dots C_k\} \in \mathbb{Z}^d$. To perform efficient inference on devices, HDC bipolarizes each class HV in AM into a bipolarized one and generate a corresponding bipolarized AM $AM^b = \{C_1^b, C_2^b, \dots C_k^b\} \in \{+1, -1\}^d$, where $C_i^b = [C_i]$.

Inference. In the inference phase, a testing data is first transformed by (1) and is encoded as a query HV. Then, HDC computes the Hamming distance between the query HV and the class HVs in the bipolarized AM. The class with the minimum distance is outputted as a prediction.

Adaptation. To enhance the classification accuracy, HDC performs adaptation by iteratively validating the training data. If the training sample is correctly classified, no change happens. However, if the query HV H of the training sample is misclassified, then H is added to the correct class HV $C_{correct}$ in AM and subtracted from the incorrectly predicted class HV C_{miss}, which can be computed as:

$$C_{correct} = C_{correct} + H, \quad C_{miss} = C_{miss} - H. \quad (3)$$

Note that we compute the Hamming distance with the AM^b but update the AM. After several iterations, the new AM^b is obtained by bipolarizing the updated AM.

3 Proposed Hyperdimensional Computing with Learnable Projection for User Adaptation Framework

In this section, we introduce two proposed techniques used in HDC for the user adaptation framework, as illustrated in Fig. 3. The first one is the learnable projection, which transforms the original HDC into a learnable network L-HDC. The architecture of L-HDC is similar to a binarized neural network (BNN) [19] since both models compute with bipolar weights. When training on cloud, L-HDC learns a better feature-aware projection by backpropagation compared with the original one described in Section 2. Then, the weights of learnable projection in L-HDC are transformed back to the learned IM and learned CIM to utilize the efficient and fast adaptability of HDC on resource-limited devices. The second one is the AM fusion mechanism to exploit the information learned from L-HDC and further improve the performance when we deploy HDC on devices. The details of these two techniques are elaborated as follows.

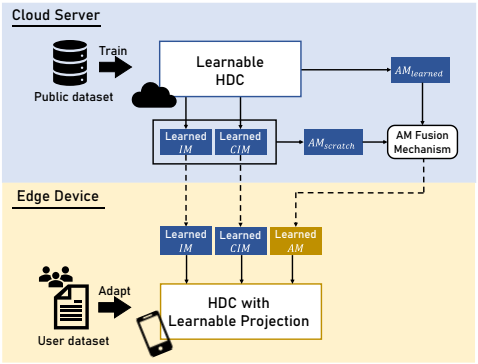


Fig. 3. Overview of the proposed HDC with the learnable projection for the user adaptation framework.

3.1 Learnable Projection

Model Architecture and Parameters. Assume that HDC is with d dimensions, m feature components, Q quantized levels, and k classes. Then, the components of IM, CIM, and AM of HDC are represented in Fig. 4(a). To transform HDC into a learnable network, L-HDC comprises two layers, as shown in Fig. 4(b). The first layer is tailored to emulate the projection and encoding parts of the original HDC. The parameters of the first layer contain two sets of weights denoted as $W^{IM} \in \{+1, -1\}^{m \times d}$ and $W^{CIM} \in \{+1, -1\}^{Q \times d}$ corresponding to the IM and CIM in the original HDC. The second layer is a fully connected (FC) layer whose weights represent the AM in the original HDC and are denoted as $W^{AM} \in \{+1, -1\}^{k \times d}$. In this paper, $W^{IM}[i][j]$, $W^{CIM}[i][j]$, and $W^{AM}[i][j]$ symbolize the entry of the i^{th} row and the j^{th} column of the W^{IM}, W^{CIM}, and W^{AM}, respectively.

Index of Feature Values. To facilitate the subsequent training, we find the index of each feature value by looking up its nearest quantized level and record the index in a matrix $Idx \in \mathbb{Z}^{n\times m}$, where n is the number of the training samples with m feature components. $Idx[i][j]$ symbolizes the entry of the i^{th} row and the j^{th} column of Idx.

Training with Bipolar Weights. In L-HDC, the training method used in the second layer (FC layer) is similar to that in BNN [19], which uses bipolar weights to compute gradients and accumulates the gradients on real-valued weights. However, the connection of the first layer tailored for HDC is not FC so that we cannot directly apply the training method in BNN. To solve this problem, we illustrate how to update and compute the gradients of W^{IM} and W^{CIM} as follows.

Suppose the gradient of the output of the first layer computed by the i^{th} training sample is $g \in R^d$, the gradient of W^{IM}, denoted as $g^{IM} \in R^{m\times d}$, is computed as

$$g^{IM}[j] = g \cdot W^{CIM}[\,Idx[i][j]]\,, for\ j = 1, 2, \dots, m, \tag{4}$$

where $\cdot$ is the dot product of two vectors. And the gradient of W^{CIM}, denoted as $g^{CIM} \in R^{Q\times d}$, is computed as

$$g^{CIM}[Idx[i][j]] = \sum(g \cdot W^{IM}[j])\,, for\ j = 1, 2, \dots, m. \tag{5}$$

Finally, we update W^{IM} and W^{CIM} by the gradients obtained from (4) and (5) with the updating techniques mentioned in [19].

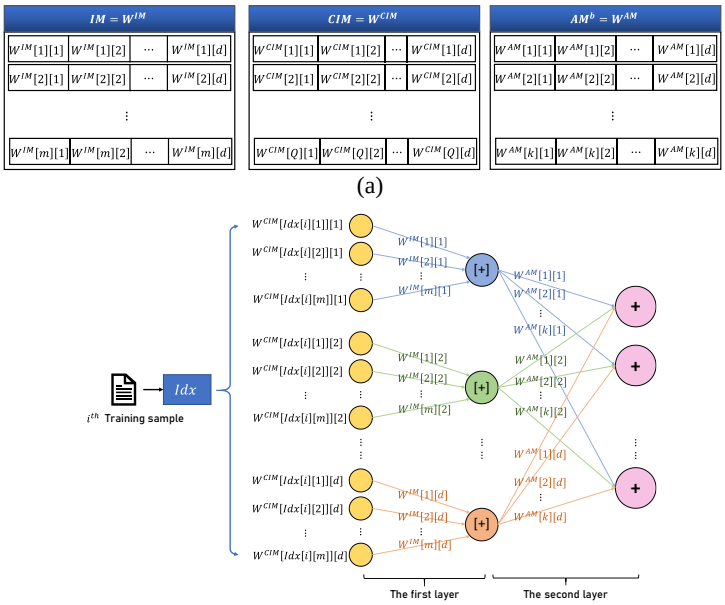


Fig. 4. (a) Demonstration of weights in L-HDC corresponding to memory tables in the original HDC. (b) The overall architecture of L-HDC.

3.2 AM Fusion Mechanism

In the training and adaptation process, L-HDC needs to iteratively compute gradients of weights and update weights by the gradient descent, which cause huge computational burdens on edge devices. Therefore, after completing training L-HDC in the cloud, we transform the architecture of L-HDC back to that of the original HDC to efficiently adapt to user's data by using (3). Note that the W^{IM} and W^{CIM} of L-HDC become the corresponding HVs in the learned IM and the learned CIM of HDC, which preserves the information of the feature correlations. However, since W^{AM} is in the bipolar representation and the adaptation of HDC relies on updating AM, whose components are integers, we cannot directly apply W^{AM} to the original HDC.

Intuitively, based on the learned IM and the learned CIM, we can obtain a new AM, denoted as $AM_{scratch}$, by training HDC from scratch. Nevertheless, the information learned in W^{AM} of L-HDC can be further exploited to improve the performance. To inherit the information in W^{AM}, we first transform W^{AM} into an AM, whose HVs corresponds to the weights of W^{AM}. Then, we multiply the AM by a scaling factor γ to generate $AM_{learned}$ with integer HVs and add $AM_{scratch}$ to $AM_{learned}$ followed by division by 2. Lastly, the combined AM is deployed to edge devices for further user adaptation. In Section 4, we demonstrate that the AM fusion mechanism which combines two types of AMs achieves higher accuracy than directly using one $AM_{scratch}$.

4 Experimental Settings and Simulation Results

4.1 Datasets and Experimental Setup

To evaluate the effectiveness of our proposed techniques for the user adaptation framework, we conduct experiments on two benchmark datasets, including the speech recognition ISOLET dataset [16] and the human activity recognition UCIHAR dataset [17]. In ISOLET, 150 subjects speak the name of 26 letters of the alphabets, and 617 features of voice signals are extracted. Each subject contains 52 data samples. As for UCIHAR, it recognizes 6 human activities based on 3-axial linear acceleration and angular velocity at a constant rate of 50Hz. There are 30 subjects in UCIHAR and each is with 343 data samples on average.

To simulate the scenario of user adaptation, we refer to the experimental settings in [5, 7]. For both datasets, we first divide them by subjects into two parts, representing the public dataset on cloud and the user dataset on edge. In practice, we randomly divide the ISOLET dataset into 100 subjects and 50 subjects as the public dataset and the user dataset, respectively. Likewise, we randomly divide UCIHAR dataset into 25 subjects and 5 subjects as the public dataset and the user dataset. Then, we further separate the public dataset into the public training dataset and the public validation dataset set with the ratio of 3:1. And the user dataset is also divided into the user adaptation dataset and the user testing dataset with the ratio of 1:1.

In the experiments, we first train L-HDC by the public training data with a learning rate of 0.001 and evaluate it with the public validation data to obtain the best model.

After training, we generate the learned IM, CIM, and the new AM by the AM fusion mechanism for the HDC model on edge. Then, the user adaptation data is used to adapt the HDC model. Finally, we utilize the user testing data to evaluate the accuracy of the HDC model. All experiments are conducted over 10 independent trials to obtain the averaged simulation results. The experimental setups are summarized in Table 1.

Table 1. Experimental setups for the user adaptation framework in (a) the ISOLET dataset and (b) the UCIHAR dataset.

Dataset	(a) ISOLET	(b) UCIHAR
# of subjects	150	30
# of classes	26	6
# of data in public training dataset	3,900	6,435
# of data in public validation dataset	1,300	2,140
# of data in user adaptation dataset	1,300	865
# of data in user testing dataset	1,300	850

4.2 Comparisons

We compare our proposed framework with the other two for user adaptation, including the original HDC framework, and the L-HDC framework without the AM fusion mechanism. All the frameworks train the model with 100 iterations on cloud and adapt the model with 50 iterations to ensure performance saturation. The details of these two compared frameworks are described as follows.

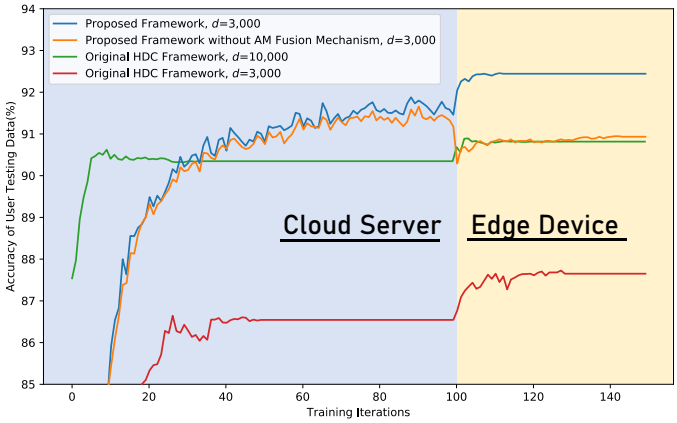
Original HDC Framework. Compared with our proposed framework, the original HDC framework trains the original HDC model on cloud and directly deploys the trained HDC model on devices for user adaptation. As shown in the simulation results, our proposed framework with the learnable projection achieves higher accuracy than the original HDC framework which ignores the feature correlations.

Our Proposed Framework without AM Fusion Mechanism. After training L-HDC on cloud, the learned IM and the learn CIM are deployed to the HDC model on edge. But in this comparison, we only transfer $AM_{scratch}$ to edge devices for user adaptation rather than that obtained by the AM fusion mechanism. The simulation results demonstrate that the AM fusion mechanism performs better than that with $AM_{scratch}$ after adaptation.

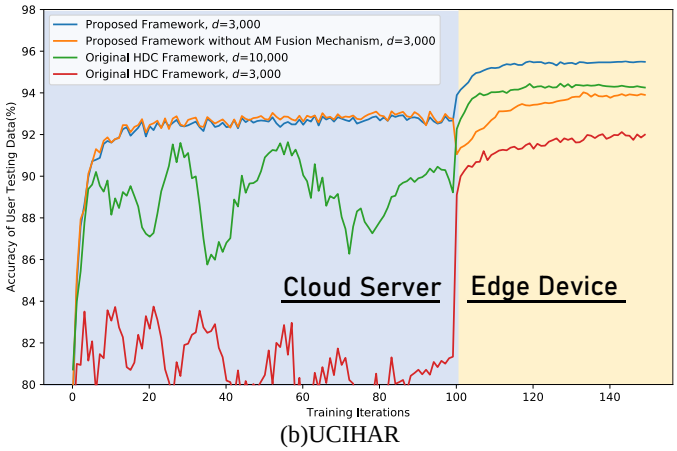
4.3 Analysis of Accuracy

Fig. 5 illustrates the accuracy curves on the user testing data of different frameworks in ISOLET and UCIHAR. Compared with the original HDC framework with d = 3,000, our proposed framework with the same dimensionality improves 4.9% and 11.3% of accuracy in ISOLET and UCIHAR, respectively, before adaptation (training

iterations=100). On the other hand, our proposed framework still provides 1.1% and 3.5% accuracy improvement compared to the original HDC framework with higher dimension d = 10,000. Both simulation results demonstrate L-HDC with the learnable projection achieves better performance than HDC with the original projection method since the feature correlations are considered. Compared with our proposed framework without the AM fusion mechanism, that with the AM fusion mechanism achieves 1.5% and 1.6% higher accuracy in ISOLET and UCIHAR, respectively, after adaptation (training iterations=150). The results show that the AM fusion mechanism can effectively preserve the knowledge learned from L-HDC and thus improve the performance. Overall, the proposed framework outperforms the original HDC framework by 4.8 % and 3.5% in ISOLET and UCIHAR, respectively. All simulation results are summarized in Table 2.



(a)ISOLET



(b)UCIHAR

Fig. 5. Accuracy analysis on the user testing data of different frameworks in (a) the ISOLET dataset and (b) the UCIHAR dataset.

Table 2. Averaged accuracy of different frameworks on the user testing data in (a) the ISOLET dataset and (b) the UCIHAR dataset

	Accuracy on User Testing Data (%)			
	(a) ISOLET		**(b) UCIHAR**	
Training iterations	**100**	**150**	**100**	**150**
Proposed framework, d=3,000	91.46	**92.44**	92.72	**95.54**
Proposed framework without AM fusion mechanism, d=3,000	91.46	90.93	92.72	93.95
Original HDC framework, d=10,000	90.34	90.81	89.26	94.3
Original HDC framework, d=3,000	86.54	87.65	81.38	92.04

4.4 Analysis of Learnable Projection

Fig. 6 illustrates the histogram of the normalized Hamming distance between each HV in the IM of the original HDC and the learned IM obtained from L-HDC, respectively. As mentioned in Section 2, HVs in IM of the original HDC are nearly orthogonal to each other. This indicates that features are uncorrelated to each other, and thus most of their mutual normalized Hamming distance is around 0.5. For the learned IM, L-HDC can learn the feature correlations during the training process. Therefore, the histogram of the learned IM is more diverse and provides HDC with a better projection for higher performance, as shown in Section 4.3.

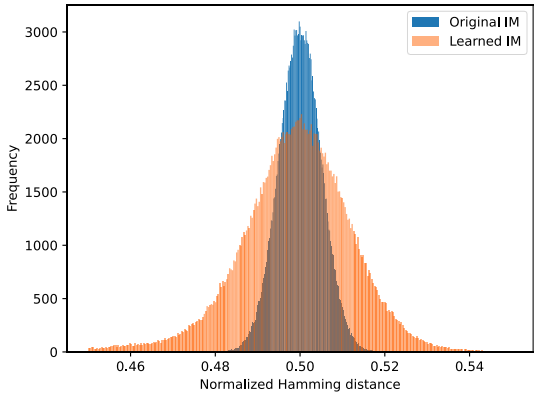


Fig. 6. Histogram of the normalized Hamming distance between each two HVs in the learned IM (orange) and the original IM (blue)

5 Conclusions

In this paper, we propose two techniques, learnable projection and AM fusion mechanism, to improve the performance of HDC in the user adaptation framework. We transform the architecture of the original HDC into L-HDC to learn the feature correlations of data. Moreover, the AM fusion mechanism exploits the AM obtained from L-HDC to further improve HDC model's performance.

Based on the simulation results, L-HDC gives each deployed model a better initial point than the original HDC in terms of accuracy for 4.9% and 11.3% improvement in ISOLET and UCIHAR, respectively. Moreover, AM fusion mechanism avoids the accuracy degradation and enhances the accuracy progress on edge devices. Overall, our proposed framework compared with the original HDC framework improves 4.8% and 3.5% of accuracy in ISOLET and UCIHAR, respectively.

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

References

1. Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M.: Internet of Things (IoT): A vision, architectural elements, and future directions. Future generation computer systems 29(7), 1645–1660 (2013).
2. Song, M., Zhong, K., Zhang, J., Hu, Y., Liu, D., Zhang, W., ... & Li, T.: In-situ ai: Towards autonomous and incremental deep learning for iot systems. In: IEEE International Symposium on High Performance Computer Architecture (HPCA), pp. 92–103. IEEE, Vienna, Austria (2018).
3. Rahimi, A., Kanerva, P., Benini, L., & Rabaey, J. M.: Efficient biosignal processing using hyperdimensional computing: Network templates for combined learning and classification of exg signals. Proceedings of the IEEE 107(1), 123–143 (2019).
4. He, J., Zhang, D., Jiang, N., Sheng, X., Farina, D., & Zhu, X.: User adaptation in long-term, open-loop myoelectric training: implications for EMG pattern recognition in prosthesis control. Journal of neural engineering 12(4), 046005 (2015).
5. Choi, S., Sim, J., Kang, M., Choi, Y., Kim, H., & Kim, L. S.: An Energy-Efficient Deep Convolutional Neural Network Training Accelerator for In Situ Personalization on Smart Devices. IEEE Journal of Solid-State Circuits 55(10), 2691–2702 (2020).
6. Lange, M. D., Jia, X., Parisot, S., Leonardis, A., Slabaugh, G., & Tuytelaars, T.: Unsupervised model personalization while preserving privacy and scalability: An open problem. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 14463–14472. (2020).
7. Matsui, S., Inoue, N., Akagi, Y., Nagino, G., & Shinoda, K.: User adaptation of convolutional neural network for human activity recognition. In: 25th European Signal Processing Conference (EUSIPCO), pp. 753–757. IEEE, Kos (2017).
8. Satyanarayanan, M.: The emergence of edge computing. Computer 50(1), 30–39 (2017).
9. Huang, S. A., Chang, K. C., Liou, H. H., & Yang, C. H.: A 1.9-mw svm processor with on-chip active learning for epileptic seizure control. IEEE Journal of Solid-State Circuits 55(2), 452–464 (2019).

10. Choi, S., Shin, J., Choi, Y., & Kim, L. S.: An optimized design technique of low-bit neural network training for personalization on IoT devices. In: Proceedings of the 56th Annual Design Automation Conference, pp. 1–6. (2019).
11. Kanerva, P.: Hyperdimensional computing: an introduction to computing in distributed representation with high-dimensional random vectors. Cognitive computation 1(2), 139–159 (2009).
12. Chang, C. Y., Chuang, Y. C., & Wu, A. Y. A.: Task-projected hyperdimensional computing for multi-task learning. In: IFIP International Conference on Artificial Intelligence Applications and Innovations, pp. 241–251. Springer, Cham, (2020).
13. Y. -C. Chuang, C. -Y. Chang and A. -Y. A. Wu.: Dynamic Hyperdimensional Computing for Improving Accuracy-Energy Efficiency Trade-Offs. In: IEEE Workshop on Signal Processing Systems (SiPS), pp. 1–5. IEEE, Coimbra, Portugal (2020).
14. C. -Y. Chang, Y. -C. Chuang and A. -Y. A. Wu.: IP-HDC: Information-Preserved Hyperdimensional Computing for Multi-Task Learning. In: IEEE Workshop on Signal Processing Systems (SiPS), pp. 1–6. IEEE, Coimbra, Portugal (2020).
15. Chang, E.J., Rahimi, A., Benini, L., Wu, A.Y.: Hyperdimensional computing-based multimodality emotion recognition with physiological signals. In: IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS), pp. 137–141. IEEE, Hsinchu, Taiwan (2019).
16. Uci machine learning repository, http://archive.ics.uci.edu/ml/datasets/ISOLET, last accessed 2021/03/01.
17. Anguita, D., Ghio, A., Oneto, L., Parra, X., & Reyes-Ortiz, J. L.: A public domain dataset for human activity recognition using smartphones. In: European Symposium on Artificial Neural Networks, pp.1–3. (2013).
18. Plate, T.A.: Holographic reduced representations. IEEE Trans. Neural Netw. 6(3), 623–641 (1995)
19. Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., & Bengio, Y.: Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. arXiv preprint arXiv:1602.02830 (2016).