EnHDC: Ensemble Learning for Brain-Inspired Hyperdimensional Computing

Ruixuan Wang, Dongning Ma, Xun Jiao
{rwang8, dma2, xun.jiao}@villanova.edu
Villanova University, Villanova, PA 19085

***Abstract*—Recently, brain-inspired hyperdimensional computing (HDC) becomes an emerging computational scheme that has achieved success in various domains such as human activity recognition, voice recognition, and bio-medical signal classification. HDC mimics the brain cognition and leverages high-dimensional vectors (e.g., 10000 dimensions) with fully distributed holographic representation and (pseudo-)randomness. Ensemble learning is a classical learning method utilizing a group of weak learners to form a strong learner, which aims to increase the accuracy of the model. This paper presents the first effort in exploring ensemble learning in the context of HDC and proposes the first ensemble HDC model referred to as `EnHDC`. `EnHDC` uses a majority voting-based mechanism to synergistically integrate the prediction outcomes of multiple base HDC classifiers. To enhance the diversity of base classifiers, we vary the encoding mechanisms, dimensions, and data width settings among base classifiers. By applying `EnHDC` on a wide range of applications, results show that `EnHDC` can achieve on average 3.2% accuracy improvement over a single HDC classifier. Further, we show that `EnHDC` with reduced dimensionality can achieve similar or even surpass the accuracy of baseline HDC with higher dimensionality. This leads to a 20% reduction of storage requirement of HDC model, which can enhance the efficiency of HDC enabled on low-power computing platforms.**

I. Introduction

Inspired by how human brain functions, hyperdimensional computing (HDC) is an emerging computing scheme that leverages the abstract patterns and mathematical properties of vectors in high dimension spaces [14], [13]. Rather than processing actual numbers, HDC works with hypervectors (HV), which are high dimensional (e.g., 10000 dimensions), holographic (not micro-coded) vectors with i.i.d. (independent and identically distributed) elements [9]. As a novel computing scheme, HDC has shown promising performance for various applications such as natural language processing [13], voice recognition [7], and bioinformatics [12]. Compared with traditional computing schemes such as neural networks, HDC has several advantages such as smaller model and lower computing cost, making it a promising computing scheme with low power computing platforms and edge computing devices [8]. In addition, the memory-centricity of HDC grants the advantage of embracing the emerging energy-efficient in-memory computing schemes over other machine learning algorithms such as neural networks [10].

Ensemble learning is a machine learning paradigm where multiple models (often called “weak learners”) are trained to solve the same problem and combined to get better results [4]. Typically, an ensemble learning system aims to improves the performance by combining diverse weak learners (base classifiers). Using an ensemble model that combines the output from several models, e.g., averaging them, can reduce the risk of an unfortunate selection of a particularly poor classifier. For HDC, prior works inspire that a collection of HDC classifiers can achieve comparable performance for specific (e.g., EEG) applications with low overhead [3]. However, such method still relies on external architectures such as linear layers that requires additional training effort for the aggregation from the results output by single HDC model. Building an ensemble classifier using HDC itself under different configurations and (hyper-)parameters as well as the design exploration still require further study.

This paper explores the use of ensemble learning on HDC models and develop the ensemble HDC classifier, **`EnHDC`** for a wide range of applications. Leveraging the diversity of base classifiers, the ensemble classifier formulated by **`EnHDC`** achieves improved accuracy and reduced model size without additional architectures such as post-processing linear layers that requires training. In particular, we make the following contributions:

- By leveraging the aggregated intelligence of a variety of HDC classifiers with different random initialization, **`EnHDC`** is able to achieve on average 3.2% accuracy improvement over a single HDC classifier.
- We diversify of base classifiers by varying a diverse set of parameters such as number of dimensions, data width, and encoding methods. This further leads to 1.2% accuracy improvement over basic **`EnHDC`** classifier.
- We evaluate the **`EnHDC`** on four different practical application domains including image classification, human activity recognition, speech recognition, and medical diagnosis. **`EnHDC`** enables smaller HDC models of about 20% size reduction with no accuracy drop.

II. HDC Preliminaries

A. Notions in HDC

Hypervector (HV) is a type of high-dimensional, holographic vectors with i.i.d. elements [9]. An HV of n dimensions can be noted by $\vec{H} = \langle h_1, h_2, \ldots, h_n \rangle$, where h_i denotes the elements inside the HV. HVs use their high dimensional space to store different layers of information, thus can represent values, features and even data samples. To establish the dynamic connection between different layers of information in HV, methodologies of aggregating information from HVs such as HDC operations, are therefore necessary.

HVs support three basic operations, **addition** ($+$), **multiplication** ($*$) and **permutation** (ρ). Additions and multiplications take two HVs as input and perform **element-wise** operations that add or multiply each element inside the HVs index by index. Permutations only take one HV and perform **cyclic shift** over the HV. For all the three operations, the input HVs and the output HVs are in the same dimension. Addition is used to aggregate parallel features that usually belongs to one modality, while multiplication is used to combine different types of features together to create new features. Permutation is used to reflect spatial or temporal changes in the features.

Similarity check is used in HDC for the objective of measuring the similarity δ of information between different HVs. There are different algorithms to measure similarity such as Euclidean distance and Hamming distance, while in **EnHDC**, we are using cosine similarity as noted by $\delta(\vec{H_p}, \vec{H_q}) = \frac{\vec{H_p} \cdot \vec{H_q}}{||\vec{H_p}|| \times ||\vec{H_q}||}$. A higher similarity between two HVs indicates that they share more alike information, or vice versa.

B. HDC in Learning Tasks

HDC in learning tasks features three major phases: **Encoding**, **Training**, and **Inference**.

Encoding is the process of mapping input features of one sample to the high-dimensional space available for HDC training, inference, i.e., building representative HVs of a sample from the fundamental item memory using combinations of HDC operations. Item memory is a type of specially allocated memory during runtime, which stores the bottom layer HVs that are used to establish other HVs. To ensure the i.i.d. property, HVs in the item memories are all randomly initialized. Assume we have the m-dimensional input features $\vec{F} = \langle f_1, f_2, \ldots, f_m \rangle$ for each sample, a set of corresponding item memories $\mathcal{R} = \{\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_m\}$ and the combination of HDC operations E determined by the application, the encoded HV $\vec{H}$ is obtained by looking up each feature's corresponding HV in the item memory and then applying them into the HDC operation combination: $\vec{H} = E(\mathcal{R}, \vec{F}) = E(\mathcal{R}_1[f_1], \mathcal{R}_2[f_2], \ldots, \mathcal{R}_m[f_m])$. The encoded HV will subsequently represent the input sample in training and inference.

Training is the process of aggregating encoded HVs sharing the same label to build the associative memory. Associative memory stores the class HVs, each representing a class in the learning problem. Training can be denoted as $\mathcal{A} = \{\sum \vec{H^1}, \sum \vec{H^2}, \ldots, \sum \vec{H^k}\}$. Assume we have a learning problem with k classes, and the encoded HVs $\vec{H^l}$ for each training sample where l means the class label, training process to establish the associative memory $\mathcal{A}$ is by summing HVs representing samples from the same class in the training set.

Inference is the process of using the associative memory established in the training phase to determine the class label of an unknown sample. Inference can be denoted as $l = argmax(\{\delta(\vec{H_q}, \vec{A^1}), \delta(\vec{H_q}, \vec{A^2}), \ldots, \delta(\vec{H_q}, \vec{A^k})\})$. First, we encode the unknown sample into its representing HV $\vec{H_q}$ referred to as the query HV. Then we perform similarity check between the query HV and each class HV inside the associative

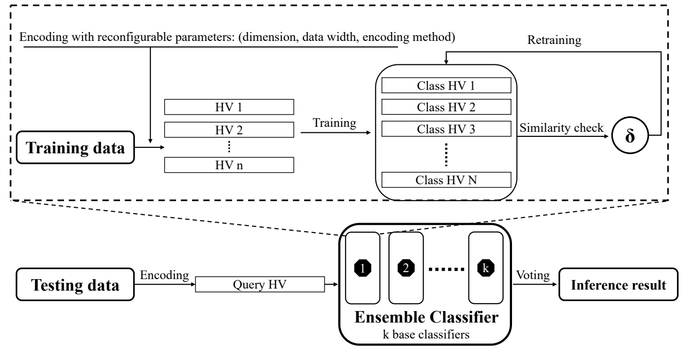

Fig. 1. **EnHDC** framework

memory. As aforementioned, higher similarity means higher common information shared by the two HVs, further indicating that these two HVs are likely from the same class. Therefore, the class of HVs in the associative memory having the highest similarity is determined to be the class of the query HV, namely the predicted label of the sample.

III. EnHDC Model

In this section, we describe the development of **EnHDC** and the enhancement to the diversity of the base classifiers in **EnHDC** for further performance improvement. Fig. 1 is the overview of **EnHDC**. First, we separately train several different base classifiers using different parameter configurations. We subsequently integrate these base classifiers into one ensemble classifier. Then, we encode the testing sample into query HV and perform inference on every base classifier. As we collect all the base inference results, we employ majority voting to obtain the inference result of the ensemble classifier.

A. Base Classifier Development

In traditional ensemble learning, base classifiers are developed with different initialization settings. In HDC, base classifiers are developed as described in Section II-B. Each base classifier has different configurations and different randomly generated item memories $\mathcal{R}$. Thus, the training outcome, i.e., the associate memory, will have different class HVs representing the same class. Therefore, for a given query input, the classification output may be different.

B. Diversity Enhancement of Base Classifier

As performance of base classifiers can be sub-par, we propose to formulate the ensemble classifier by diversifying the base classifiers with different configurations including encoding mechanisms, dimensions, and data widths. Specifically, we use two set of encoding mechanisms: the Record based encoding and the N-gram based encoding.

Record encoding is a general encoding method which maps every feature vector $\vec{F} = \langle f_1, f_2, \ldots, f_m \rangle$, into hyperdimensional space. It finds the minimum and maximum feature values and projects the range into p feature levels. A set of random and orthogonal bipolar HVs $\vec{H_l}$ is assigned to every feature level. Meanwhile, for preserving the position independence of feature values in the feature vector, Record encoding method also assigns one set of HVs to each feature values, referred to as the base HVs $\vec{H}_b$. The Record encoding method uses the level HVs to represent each feature value in the feature vector and base HVs for the position relationship of features values. Record encoding is employed by linearly combining the level HVs and base HVs: $\vec{H}_{Record} = \sum_{i=1}^{m} \vec{H}_{li} * \vec{H}_{bi}$. $\vec{H}_{Record}$ is the non-bipolar encoded HV with D dimensions containing integer values. Meanwhile, since base HVs are randomly generated, they are almost mutually orthogonal, which means the cosine similarity between any two base HVs $\delta(\vec{H}_{bi}, \vec{H}_{bj})$ approximately equals to 0.

We also use the **N-gram encoding** method, in which we employ the locality-based sparse random projection [6] as our method. If we are using the D dimensional HVs, we first extended the length of feature vector from N to D. For instance, in **MNIST** dataset, when we try to encode a feature vector with N = 768 feature (pixel) values into D = 10000 dimension HVs, we firstly need to attach 13 duplication following the original feature vector. Meanwhile, we generate a random bipolar D dimensional local-hashing HV $\vec{H}_s = \langle h_{s1}, h_{s2}, \ldots, h_{sn} \rangle$. To encode the extended feature vector, N-gram encoding method deploys an N-gram sliding window and takes the dot product of the extended feature vector and projection vector in this window range: $\vec{H}_{N-gram} = \langle v_1, v_2, \ldots, v_n \rangle$. The i-th value of $\vec{H}_{N-gram}$ equals to the dot product of the w feature values from f_i to f_{i+w-1} and w element values from h_i to h_{i+w-1}, where w is the size of sliding window.

To represent the internal data of the HDC base classifiers, we use three data widths: INT_8 and INT_16 and three different dimension settings: 1000, 5000, and 10000 for base classifiers to enhance the diversity of base classifiers.

C. Voting Mechanism

The inference phase has two steps in the **EnHDC**. First, for every base classifier, we map each testing data into a query HV $\vec{H}_q$ using the same encoding method during training and calculate the cosine similarity of each class HVs with the query HV $\vec{H}_q$ in every base classifier. The inference result is pointed to the class with the highest cosine similarity. Second, we collect all the base classifier inference results in the ensemble classifier to vote the ensemble inference result.

We explored two voting mechanisms to get a better inference result and we tested two different voting strategies: soft voting and hard voting. Hard voting is the majority voting, while for soft voting, since each base classifier gave the inference result by cosine similarity checking, we can sum up all the related cosine distances and rank them in order, where the champion will be selected as the final result. Our test result shows that the hard voting strategy achieves better accuracy in **EnHDC**. Therefore, we integrate all the base inference results in ensemble classifier and use majority voting to get the ensemble inference result of the corresponding query HV.

IV. Experimental Results

We evaluate **EnHDC** using four application domains: speech recognition (**ISOLET** [5]), human activity recognition (**HAR** [1]), handwritten digits (**MNIST** [11]), and cardiotocography (**CARDIO** [5]).

A. Accuracy Improvement

Fig. 2 and 3 present the accuracy comparison between **EnHDC** configurations. The baseline has one HDC classifier while the **EnHDC** employs several different base classifiers.

We can observe in Fig. 2 that, **EnHDC** contains 8 and 16 base classifiers with different encoding methods (Record and N-gram encoding). To evaluate the performance of **EnHDC**, we compare 3 models with the dimensionality setting across $D = 1000, 5000, 10000$, **EnHDC** is showing consistent higher accuracy than baseline models. When **EnHDC** has 8 base classifiers, the average improvement is 3.2%. Normally HDC requires a high dimensionality such as $D = 10000$ to achieve satisfying performance. However, with **EnHDC**, we can even achieve higher accuracy with lower dimension. Across all the applications, **EnHDC** with 8 classifiers under $D = 1000$ presents higher accuracy than baseline model under $D = 10000$. The average improvement is 1.37%.

Additionally, as shown in Fig. 3, the number of base classifiers have a notable impact on the accuracy. Without the loss of generality, our experiment features different ensemble sizes, starting from two base classifiers to twelve base classifiers. As the green line shown in Fig. 3, the accuracy of **EnHDC** increases by adding more classifiers but comes to the vertex when using eight base classifiers for most applications. After this, the accuracy improvement is saturated. This is consistent with the ensemble theory in [2], where the performance of ensemble learning algorithms cannot constantly increase by adding an infinite amount of base classifiers. The accuracy will peak during the progress of increasing the number of classifiers, and after this peak value, the overall accuracy cannot have an obvious improvement.

The diversity enhancement can further improve the accuracy of **EnHDC** as shown in Fig. 3. **EnHDC**_enhanced classifier is the **EnHDC** classifier with enhanced diversity by varying encoding mechanisms (Record encoding, N-gram encoding), dimensions ($D = 1000, 5000, 10000$), and data width (INT_8, INT_16). This figure shows that the **EnHDC**_enhanced classifier can further improve the accuracy of **EnHDC** classifier by 1.2% on average across all applications, which is 4.4% improvement over baseline HDC model.

B. Model Size Reduction

Typically, HDC is required to have a high dimensionality, e.g., 10000, to achieve a satisfying performance. For example, for **CARDIO** dataset in Fig. 2, the baseline HDC classifier with 10000 dimensions has 1.9% higher accuracy than the baseline HDC classifier with 1000 dimensions. However, with ensemble learning, we can see that **EnHDC** with just 1000 dimensions is able to achieve similar level or even surpass the accuracy of 10000-dimension baseline HDC classifiers across all applications. This can achieve a reduction on the HDC model size. For example, with 8 base classifiers with $D = 1000$ dimensions, this can reduce 20% model size compared to a baseline classifier with $D = 10000$ dimensions. For **MNIST** dataset with 10 classes, we have a baseline HDC model with $D = 10000$ and INT_8 data width whose model size is 8 bits $\times$ 10000 dimensions $\times$ 10 classes = 800Kb

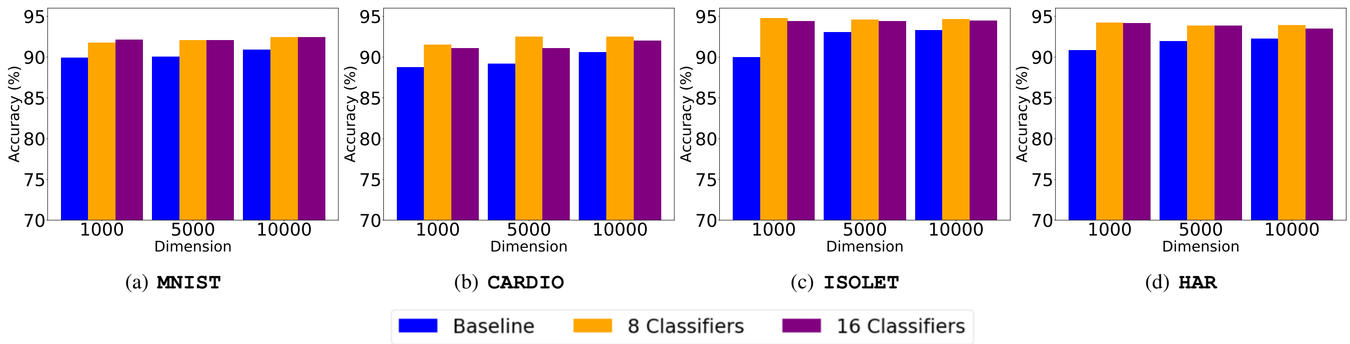

Fig. 2. **EnHDC** performance under different HV dimension: D = 1000, 5000, 10000

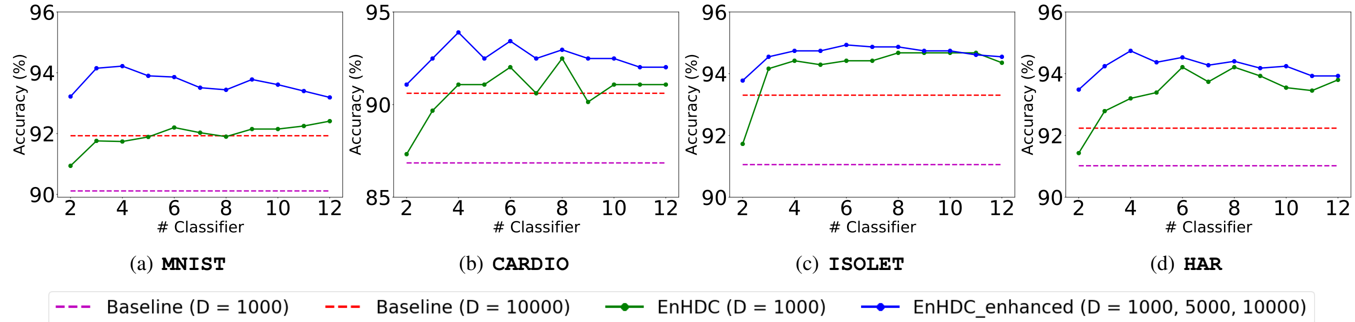

Fig. 3. **EnHDC** performance under different ensemble size

and **EnHDC** with base classifiers with $D = 1000$ and INT_8 whose model size is 8 bits $\times$ 1000 dimensions $\times$ 10 classes $\times$ 8 classifiers = 640Kb. The overall model size reduction is 160Kb for **MNIST**. Meanwhile, in **HAR** dataset with 12 classes, we have one 960Kb baseline model with $D = 10000$ and **EnHDC** with eight 96Kb base classifiers with $D = 1000$, pointing out the model size reduction for **HAR** is 180Kb.

V. Conclusion

This paper presents **EnHDC**, the first ensemble classifier to the best of our knowledge for the brain inspired hyperdimensional computing. **EnHDC** employs different base classifiers under different HV dimensions, data widths, and encoding methods. **EnHDC** applies the majority voting to generate the final inference result from base classifiers that are individually trained. By evaluating on four applications, we show that **EnHDC** can achieve higher accuracy and can reduce model size compared to baseline HDC classifiers. Additionally, by increasing the diversity of base classifiers, the classification accuracy has an enhanced improvement compared to the original **EnHDC** model. This paper presents effort in using an ensemble learning in HDC for boosting the performance and can enhance it for implementing in low power platforms such as edge computing architectures and embedded systems.

References

[1] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *Esann*, volume 3, page 3, 2013.

[2] H. Bonab and F. Can. Less is more: A comprehensive framework for the number of components of ensemble classifiers. *IEEE Transactions on Neural Networks and Learning Systems*, 30(9):2735–2745, 2019.

[3] Alessio Burrello, Simone Benatti, Kaspar Schindler, Luca Benini, and Abbas Rahimi. An ensemble of hyperdimensional classifiers: hardware-friendly short-latency seizure detection with automatic ieeg electrode selection. *IEEE journal of biomedical and health informatics*, 25(4):935–946, 2020.

[4] Xibin Dong, Zhiwen Yu, Wenming Cao, Yifan Shi, and Qianli Ma. A survey on ensemble learning. *Frontiers of Computer Science*, 14(2):241–258, 2020.

[5] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[6] M. Imani, J. Morris, J. Messerly, H. Shu, Y. Deng, and T. Rosing. Bric: Locality-based encoding for energy-efficient brain-inspired hyperdimensional computing. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2019.

[7] Mohsen Imani, Deqian Kong, Abbas Rahimi, and Tajana Rosing. Voicehd: Hyperdimensional computing for efficient speech recognition. In *2017 IEEE International Conference on Rebooting Computing (ICRC)*, pages 1–8. IEEE, 2017.

[8] Mohsen Imani, Xunzhao Yin, John Messerly, Saransh Gupta, Michael Niemier, Xiaobo Sharon Hu, and Tajana Rosing. Searchd: A memory-centric hyperdimensional computing with stochastic training. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(10):2422–2433, 2019.

[9] Pentti Kanerva. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive computation*, 1(2):139–159, 2009.

[10] Geethan Karunaratne, Manuel Le Gallo, Giovanni Cherubini, Luca Benini, Abbas Rahimi, and Abu Sebastian. In-memory hyperdimensional computing. *Nature Electronics*, 3(6):327–337, 2020.

[11] Yann LeCun et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[12] Dongning Ma, Rahul Thapa, and Xun Jiao. Molehd: Drug discovery using brain-inspired hyperdimensional computing. *arXiv preprint arXiv:2106.02894*, 2021.

[13] Fateme Rasti Najafabadi, Abbas Rahimi, Pentti Kanerva, and Jan M Rabaey. Hyperdimensional computing for text classification. In *Design, Automation Test in Europe Conference Exhibition (DATE), University Booth*, pages 1–1, 2016.

[14] Abbas Rahimi, Simone Benatti, Pentti Kanerva, Luca Benini, and Jan M Rabaey. Hyperdimensional biosignal processing: A case study for emg-based hand gesture recognition. In *2016 IEEE International Conference on Rebooting Computing (ICRC)*, pages 1–8. IEEE, 2016.