IP-HDC: Information-Preserved Hyperdimensional Computing for Multi-task Learning

Cheng-Yang Chang, Yu-Chuan Chuang, and An-Yeu (Andy) Wu, Fellow, IEEE
Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan
{kevin, frankchuang}@access.ee.ntu.edu.tw, andyw@ntu.edu.tw

Abstract—Brain-inspired Hyperdimensional (HD) computing has shown great success in many real-world applications requiring low-power designs, e.g., edge computing for Internet of Things. For edge computing, multi-task learning (MTL) is more desirable than single-task learning due to more efficient model deployments for resource-constrained devices. As a lightweight algorithm, HD computing is compatible with MTL. However, despite its energy-efficiency, the memory overhead of HD computing significantly increases with the number of tasks under the MTL scenario. In this paper, we propose Information-Preserved HD computing (IP-HDC) to make the HD model simultaneously support multiple tasks with negligible memory overhead. Moreover, to mitigate interference between tasks, we introduce “mask” HD vectors by extracting the informative dimensions of each task. Compared with the baseline method, IP-HDC shows a 22.9% of accuracy improvement and demonstrates both reliability and scalability by efficiently achieving isolation between tasks in the HD space.

Index Terms—Brain-inspired computing, Hyperdimensional computing, Multi-task learning, Model compression, Memory efficiency

I. INTRODUCTION

With the rise of data explosion, Internet of Things (IoT) manages to collect and analyze a huge amount of data with robust learning algorithms on edge. However, due to limited resources available on edge devices, numerous advantages brought by IoT are feasible only if the trade-off between model performance and hardware efficiency is well studied [1]. Meanwhile, multi-task learning (MTL) is grabbing attention, since a machine learning model that can accommodate multiple cognitive tasks is more desirable for the future of IoT [2].

Brain-inspired Hyperdimensional (HD) computing [3] is emerging as a lightweight alternative classifier to the high-complexity deep neural network (DNN). The core idea of HD computing involves emulating neural activities by transforming information into HD vectors with very high (e.g. thousands) dimensionality. With the well-defined arithmetic operators on HD vectors, HD computing has shown successful progress in many real-world applications [4]-[7]. Moreover, compared with conventional learning algorithms, HD computing is suitable for parallel architecture. Several FPGA-accelerated and ultra-low-power implementations have been proposed based on its massively parallel operations [8]-[10]. Larger energy saving is further achieved by using emerging 3D nanoscale devices [11]. Other leading properties of HD computing include fast convergence without complicated backpropagation [12] and robustness against hardware failures [13]-[14].

HD computing learns to tackle cognitive tasks by retrieving discriminative information and storing the corresponding HD vectors in the associative memory (AM). However, the size of AM grows linearly with the number of tasks under the MTL scenario. The huge memory overhead brought by AM becomes a significant problem and hinders HD computing from the practical deployment of the MTL system on edge devices. Some methods have been proposed to reduce the size of AM. First, HD vectors belonging to different tasks can be sequentially averaged into one representative HD vector for all tasks. However, this approach confronts unacceptable performance degradation due to the severe interference between tasks. To overcome interference between tasks, [15] is proposed to use the technique of random projection to retain information of each task. However, this work focuses on binarized HD vectors, which cannot provide acceptable performance on practical applications [16]-[17]. Besides, the statistical information stored in AM cannot be fully exploited due to the lack of analytical projection vectors. Therefore, we need to investigate more systematical and analytical approaches instead.

In this paper, we propose a novel framework of Information-Preserved Hyperdimensional Computing (IP-HDC) to make HD computing simultaneously support multiple tasks. Instead of using binarized HD vectors, we use real-valued HD elements to provide satisfactory classification accuracy. The core idea of IP-HDC comes from over-parameterization in DNN, which implies that only a small subspace spanned by the optimal parameters is occupied by the given task [18]. We observe that a similar phenomenon occurs in HD computing, called dimension-redundancy, which means that only a small portion of dimensions has large impacts on classification result. Hence, in IP-HDC, we only extract informative dimensions for each task. After that, the “mask” HD vector of each task is introduced to mitigate the disruptive impacts of interference between tasks. In the inference phase, we utilize mask HD vector to project HD vectors of each task to the independent subspace with the most information preserved. In other words, mask HD vectors help split the HD space into subspaces for different tasks with minimal interference between them. Compared with the baseline method, IP-HDC efficiently achieves isolation between tasks in the HD space and shows a 22.9% of improvement in terms of classification accuracy with negligible memory overhead.

The rest of this paper is organized as follows. Section II presents an overview of HD computing. In Section III, we introduce the framework and techniques of the proposed IP-
HDC. The experimental results are discussed in Section IV. Finally, we conclude this paper in Section V.

II. PRELIMINARIES

A. Hyperdimensional Computing [3]

In this subsection, we review the processing flow of a general HD model, as shown in Fig. 1.

**Nonlinear Mapping:** In HD computing, a feature vector with \( m \) feature components \( f \in \mathbb{R}^m \) would be nonlinearily transformed into HD vectors of \( D \) dimensionality with equally probable (-1)s and 1s. HD computing starts by constructing the Item Memory (IM) and the Continuous item Memory (CiM). IM = \( \{ID_1, ID_2, \ldots, ID_n\} \), where \( ID_i \in (-1,1)^D \) corresponds to the HD vector of the \( i \)th feature component. HD vectors in IM are mutually orthogonal for random projection. CiM = \( \{L_1, L_2, \ldots, L_n\} \), where \( L_j \in (-1,1)^D \) corresponds to the HD vector of the \( j \)th quantization level of actual feature values. To construct the CiM, we first find the maximum value \( V_{max} \) and minimum value \( V_{min} \) of each feature component. The range between \( V_{max} \) and \( V_{min} \) is equally quantized to \( \ell \) levels, and two random HD vectors \( L_j, L_j \) are assigned to \( V_{max} \) and \( V_{min} \), respectively. To preserve spatial relation between quantization levels, other values of the specific feature component are associated with HD vectors whose Hamming distance are proportionate to \( L_j \) and \( L_\ell \). Therefore, nonlinear mapping of a feature vector \( f \in \mathbb{R}^m \) comprises discretization and looking up the corresponding vectors \( \{S_1, S_2, \ldots, S_m\} \) from the CiM. After mapping, a set of two-vector pairs \( I = \{(S_1, ID_1), (S_2, ID_2), \ldots, (S_m, ID_m)\} \) is generated.

**Encoding:** In HD computing, two arithmetic operators are defined to perform encoding. One is binding, dimension-wise XOR operation \( \oplus \) between two HD vectors. The other one is bundling, dimension-wise addition + between two HD vectors. Binding creates a dissimilar HD vector to the corresponding inputs, which is suitable for projection. Bundling produces an HD vector that is similar to the corresponding inputs, which is suitable for representing sets. Based on these two operators followed by binarization function \( [\cdot] \) defined in equation (2), HD model encodes pairs of vectors in \( I \) into a representative HD vector \( H \in (-1,1)^D \), which can be expressed as

\[
H = \left[ \sum_{i=1}^{m} S_i \oplus ID_i \right].
\] (1)

For a \( k \) -class classification task, a set of class HD vectors \( \{M^1, M^2, \ldots, M^k\} \) are stored in the associative memory (AM). Finally, all class HD vectors are normalized to the unit norm for simplifying the computation of the cosine similarity to the inner product between HD vectors. In the inference phase, an unseen testing data undergoes the same nonlinear mapping and encoding as in the training phase, and the data is encoded as a query vector \( Q \in (-1,1)^D \) by equation (1). The HD model calculates the inner products between \( Q \) and all class HD vectors \( \{M^1, M^2, \ldots, M^k\} \) stored in the AM. The class with the highest inner product is selected as the classification result, which can be expressed as

\[
\text{Classification Result} = \arg\max_j \langle Q \cdot M^j \rangle,
\] (4)

where \( \cdot \) denotes the inner product.

B. Multi-task Learning for HD Computing

Before diving into the illustration of the proposed framework, we introduce the scenario of the HD computing-based MTL system. Assume that there are \( s \) tasks under the MTL scenario, and each of them contains \( k \) classes. An AM table would be employed to store \( s \) independent AMs for \( s \) tasks, as shown in Fig. 2, where \( M'_j \) represents the \( j \)th class HD vector of the \( i \)th task. However, the memory overhead of the AM table increases with the number of tasks, which is unacceptable for resource-constrained edge devices. Hence, we can formulate the problem of the HD computing-based MTL system as compressing the AM table to a single AM whose size is independent of the number of tasks while preserving the most information for each task.
III. PROPOSED INFORMATION-PRESERVED HYPERDIMENSIONAL COMPUTING (IP-HDC)

The proposed framework of IP-HDC is illustrated in Fig. 3(a). IP-HDC consists of two stages: 1) the off-line training stage and 2) the on-line inference stage. In the off-line training stage, we evaluate the importance of each dimension and introduce “mask” HD vectors to effectively compress the original AM table into a single AM while preserving as much information as possible. In the on-line inference stage, we utilize the mask HD vector to project data into the subspace where other tasks in the MTL system impose minimal interference. In the following subsections, we investigate the two stages in detail.

A. Off-line Training Stage

1) Dimension Ranking and Mask HD Vectors

Given \( s \) tasks under the MTL scenario, and each of them contains \( k \) classes, then the original AM table comprises \( s \times k \) class HD vectors. \( M^1_i \) denotes the \( i^{th} \) class HD vector of the \( i^{th} \) task \( T_i \) for \( i = 1, 2, \ldots, s \), \( j = 1, 2, \ldots, k \). In the off-line training stage, we first rank the importance of all dimensions for each task. To evaluate the importance of each dimension to \( T_i \), the variation HD vector \( V_i \in \mathbb{R}^D \) is introduced.

\[
V_i[d] = max_{j} M^1_i[d] - min_{j} M^1_i[d], \quad \text{for } d = 1, 2, \ldots, D. \tag{5}
\]

Fig. 4 shows the distribution of the variation values when the dimensionality of HD space is set as \( D = 10000 \). We observe that, in some dimensions, all class HD vectors of \( T_i \) store noise-like information, which adds a relatively negligible difference between classes during the calculation of inner product. On the other hand, those dimensions with large variation values have the greatest impact on differentiating the classes. Hence, \( V_i \) can be considered as the information-revealing vector for \( T_i \).

Based on the variation HD vector \( V_i \in \mathbb{R}^D \), all dimensions have different degrees of importance to the classification of \( T_i \). To rank the importance, we sort \( V_i \) according to the magnitude of the variation values. After sorting, the variation HD vector is rearranged in ascending order, denoted as \( V'_i \in \mathbb{R}^D \). Hence, we can set a reservation ratio \( \gamma \) of \( 0 \leq \gamma \leq 1 \), which is used to determine the ratio of dimensions that are reserved for each task.

Based on \( \gamma \), the threshold \( \varepsilon_i \) for \( T_i \) can be determined as follows.

\[
\varepsilon_i = V'_i[(1 - \gamma) \times D], \quad \text{for } i = 1, 2, \ldots, s. \tag{6}
\]

After the determination of the threshold \( \varepsilon_i \), the mask HD vector \( K_i \in (0, 1)^D \) for the \( i^{th} \) task \( T_i \) can be derived.

\[
K_i[d] = \begin{cases} 
1 \text{ if } V'_i[d] \geq \varepsilon_i, \text{ for } d = 1, 2, \ldots, D. \\
0 \text{ if } V'_i[d] < \varepsilon_i, \text{ for } d = 1, 2, \ldots, D. 
\end{cases} \tag{7}
\]

\( K_i \) decides which dimensions are reserved for \( T_i \) to make the classification. Only those dimensions with variation values greater than the threshold \( \varepsilon_i \) would be activated in the on-line inference stage. In other words, \( K_i \) indicates the most discriminative dimensions for \( T_i \) to make the classification. Besides, the mask HD vector \( K_i \) also prevents noise-like information from interfering with the other tasks. Furthermore, mask HD vectors are stored in binary representation, which leads to negligible memory overhead to the IP-HDC system.

2) Compression of AM Table to Single AM

Given a sequence of \( s \) classification tasks \( \{T_1, T_2, \ldots, T_s\} \), and each of them contains \( k \) classes, then \( s \) mask HD vectors \( \{K_1, K_2, \ldots, K_s\} \) can be derived. To achieve isolation between tasks in the HD space, IP-HDC compresses the \( s \) class HD vectors belonging to the same \( j^{th} \) class \( \{M^1_j, M^2_j, \ldots, M^s_j\} \) with the mask HD vectors \( \{K_1, K_2, \ldots, K_s\} \), respectively, as shown in Fig. 3(b). The new class HD vector \( C^j \) of the \( j^{th} \) class can be formed by
\[ C^j = \sum_{i=1}^{s} M^j_i \odot K^i, \text{ for } j = 1, 2, \ldots, k, \quad (8) \]

where \( \odot \) represents dimension-wise multiplication. By utilizing the mask HD vectors for compression, \( k \) new class HD vectors \( \{C^1, C^2, \ldots, C^k\} \) are generated. The original AM table with the size of \( O(s \times k) \) is compressed to the single AM with the size of \( O(k) \). Meanwhile, we can mitigate the negative impact of interference between tasks after compression.

B. On-line Inference Stage

In the online inference stage, a given feature vector \( f \in \mathbb{R}^m \) of the \( t \)th task \( T_t \) is nonlinearily mapped and encoded into a query HD vector, denoted as \( Q \in (-1, 1)^D \) by equation (1). Since the single AM in Fig. 3(b) is the result of the superimposition of \( s \) tasks under the MTL scenario, IP-HDC projects \( Q \) onto the subspace belonging to \( T_t \) with the HD vector \( K_t \) formed in the off-line training stage, which is computed as

\[ \hat{Q} = Q \odot K_t. \quad (9) \]

This can be regarded as the information retrieval of the \( i \)th task \( T_i \) from the compressed AM. \( \hat{Q} \) is supposed to be particularly related to discrimination between classes of \( T_i \), which indicates that this subspace is minimally interfered with by the other tasks. Finally, the class with the highest inner product is selected as the classification result, which can be expressed as

\[ \text{Classification Result} = \arg \max_j (\hat{Q} \cdot C^j). \quad (10) \]

Note that the calculation can be skipped regarding dimensions where the values in the mask HD vector \( K_t \) are zero. In other words, this modified inner product evaluation can also reduce the computational cost of HD computing.

IV. EXPERIMENT SETTINGS AND RESULTS

A. Comparisons

We compare IP-HDC with the frameworks of the ideal benchmark and the baseline method as described in [15]. For the ideal benchmark, we consider the case where computing resources are unconstrained so that the size of the AM table is allowed to grow linearly with the number of tasks. In other words, each task has its own AM. The ideal benchmark stores the class HD vectors without any information loss, so it can be considered as the upper bound of classification accuracy. On the contrary, the baseline method naïvely averages the class HD vectors corresponding to the same class in the AM table, which causes severe interference between tasks. Therefore, the baseline method can be regarded as the lower bound of classification accuracy.

B. Experimental Settings

We validate the effectiveness of our proposed IP-HDC on a 26-class alphabet voice recognition dataset, called ISOLET [20]. In ISOLET, 150 subjects speak the name of 26 letters of the alphabets, and 617 features of voice signals are extracted, including spectral coefficients, contour features, and sonorant features. Following the concept of Split MNIST, a standard benchmark for multi-task learning [21], we design Split ISOLET. In Split ISOLET, we split 26 alphabets into \( s \) disjoint sets, each of which corresponds to a specific task in \( \{T_1, T_2, \ldots, T_s\} \).

In our experimental settings, the dimension of the HD space and quantization levels are set as \( D = 10000, \ell = 8 \), respectively, where the performance of all HD computing models saturates. To meet the expectations of the MTL scenario for reducing hardware costs, the mapping and encoding modules are shared across all tasks. All feature components are continuous, real-valued, and scaled into the range of \([−1, +1]\). We use the reservation ratio \( \gamma = 0.1 \) for stable performance of all tasks in the IP-HDC system. All experiments are conducted over 50 independent runs to obtain the averaged results.

C. Experimental Results

Robustness of IP-HDC: First, we evaluate the proposed IP-HDC with a 3-task MTL configuration. Tasks A, B, and C all contained 8 alphabets which are different from those of the other two tasks. In this case, the number of tasks \( s \) and the number of classes \( k \) are set as \( s = 3 \) and \( k = 8 \), respectively. Random partitions are conducted for 50 different independent runs to ensure the robustness of the system.

In the training stage, an AM table is constructed, as shown in Fig. 3(b). Each task is trained for 200 steps, and a training sample was randomly drawn in each step to update the corresponding class HD vector in the AM table. The mask HD vectors \( \{K_A, K_B, K_C\} \) are updated by equations (5)-(7) in each step, and they are exploited to compress the AM table to a single AM based on equation (8). To test the accuracy of the proposed IP-HDC system in each step, the compressed AM along with the mask HD vectors are used to perform classification by equations (9)-(10).

Fig. 5 illustrates the learning curves of different HD models trained on Split ISOLET. For the ideal benchmark, since no interference exists between tasks, it performs the best among the three methods at the cost of huge memory overhead. For the baseline method, the performance of task A degrades drastically in the transitions of training procedures. Although the single AM of the baseline method in the 200th training step is optimized regarding task A, the 10 training samples from task B severely scramble the single AM in the 210th training step. This leads to poor performance for tasks A and B both. When
more data from task B make the single AM converge again in the 400th training step, task A cannot retain its accuracy due to the interference from task B. The same phenomenon occurs in the transition from task B to C in the 400th training step. Furthermore, all tasks converge at a much lower accuracy level, compared with the other two methods. These phenomena stem from the fact that severe interference exists between tasks; therefore, compared with the ideal benchmark, a 26.7% of accuracy drop happens in task A. On the other hand, our proposed IP-HDC only slightly degrades classification accuracy by 3.8% on task A, compared with the ideal benchmark. Besides, the accuracy of tasks B and C are close to those of the ideal case, which only causes 3.1% and 5.4% of loss, respectively. From the simulation results, we demonstrate that IP-HDC utilizes the mask HD vectors to achieve isolation between tasks so that the information required for classification is retained with minimal degrees of interference between tasks.

**Scalability of IP-HDC:** To further ensure the scalability of IP-HDC, we verify IP-HDC on cases with varied numbers of tasks. The simulation setup is almost the same as the previous experiment except that the 26 classes are split into 13 disjoint sets in this case. Each set contains 2 classes and corresponds to a specific task. In this case, the number of tasks s and the number of classes k are set as s = 2, 3,…, 13 and k = 2, respectively. As shown in Fig. 6, although the homogeneous tasks tend to interfere with each other severely, IP-HDC can still perform close to the ideal benchmark consistently, with a slight 1.7% of accuracy drop in the 13-task case compared with the ideal benchmark. On the other hand, the baseline method suffers from the negative impacts of interference between tasks and has a much larger accuracy drop by 19.7%.

**Discussion of Reservation Ratio γ:** Here, we discuss the impact of γ in the following simulation. The simulation setup is almost the same as the previous experiment except that the number of tasks s is fixed at s = 13 in this case. IP-HDC with a higher γ would reserve more dimensions in HD space for each task. However, large γ also causes severer interference between tasks. On the other hand, small γ would lead to insufficient information reserved for each task, which may cause performance degradation. Therefore, the value of γ determines a trade-off between interference between tasks and the amount of information reserved for each task. According to the different ranges of γ in Fig. 7, we can define three regions to qualitatively describe the impact of γ: region I (γ < 0.0004) as under-masking, region II (γ = 0.0004–0.08) as effective masking, and region III (γ > 0.08) as over-masking. In the region I, the accuracy quickly decreases with lower γ, since the reserved information is not enough for a single task to make a reliable decision. In region II, the accuracy remains stable within a range of ±4.1%. In region III, the accuracy falls from 94.2% (γ = 0.08) to 78.2% (γ = 0.9). After γ = 0.5, the behavior of IP-HDC is similar to that of the baseline method, so the two methods provide nearly the same classification accuracy. Based on the experiment, we show that the reservation ratio γ is highly related to the performance of IP-HDC. Hence, to achieve an effective and accurate HD computing-based MTL system, we should analytically select the reservation ratio γ to make it locate in region II.

**V. CONCLUSION**

In this paper, we propose the Information-Preserved HD computing (IP-HDC) framework to realize an HD computing-based multi-task learning system for the future of IoT. We analyze the AM table based on the variation in each dimension and introduce a mask HD vector for each task. With the mask HD vectors, the original AM table is effectively compressed with minimal interference between tasks. Compared with the baseline method, IP-HDC shows 22.9% of accuracy improvement. Moreover, our proposed framework shows both reliability and scalability based on the experimental results.

**REFERENCES**


