MulTa-HDC: A Multi-Task Learning Framework For Hyperdimensional Computing

Cheng-Yang Chang©, Student Member, IEEE, Yu-Chuan Chuang©, Student Member, IEEE, En-Jui Chang©, Member, IEEE, and An-Yeu (Andy) Wu®, Fellow, IEEE

Abstract—Brain-inspired Hyperdimensional computing (HDC) has shown its effectiveness in low-power/energy designs for edge computing in the Internet of Things (IoT). Due to limited resources available on edge devices, multi-task learning (MTL), which accommodates multiple cognitive tasks in one model, is considered a more efficient deployment of HDC. However, as the number of tasks increases, MTL-based HDC (MTL-HDC) suffers from the huge overhead of associative memory (AM) and performance degradation. This hinders MTL-HDC from the practical realization on edge devices. This article aims to establish an MTL framework for HDC to achieve a flexible and efficient trade-off between memory overhead and performance degradation. For the shared-AM approach, we propose Dimension Ranking for Effective AM Sharing (DREAMS) to effectively merge multiple AMs while preserving as much information of each task as possible. For the independent-AM approach, we propose Dimension Ranking for Independent Memory Retrieval (DRIMER) to extract and concatenate informative components of AMs while mitigating interferences among tasks. By leveraging both mechanisms, we propose a hybrid framework of Multi-Tasking HDC, called MulTa-HDC. To adapt an MTL-HDC system to an edge device given a memory resource budget, MulTa-HDC utilizes three parameters to flexibly adjust the proportion of the shared AM and independent AMs. The proposed MulTa-HDC is widely evaluated across three common benchmarks under two standard task protocols. The simulation results of ISOLET, UCIHAR, and MNIST datasets demonstrate that the proposed MulTa-HDC outperforms other state-of-the-art compressed HD models, including SparseHD and CompHD, by up to 8.23% in terms of classification accuracy.

Index Terms—Brain-inspired computing, hyperdimensional computing, machine learning, multi-task learning

1 INTRODUCTION

With the emergence of the Internet of Things (IoT), massive data streams are produced daily. Machine learning (ML)-on-edge provides a low-bandwidth and speedy way to process the vast amounts of data on the front. This also indicates that edge devices may need to handle various tasks, such as speech recognition, human activity recognition, and image classification, in a single inference engine. Due to limited resources available on edge devices, multi-task learning (MTL), where a single ML engine simultaneously executes multiple tasks, receives much attention for future IoT applications [1]. Instead of training models separately, MTL aims to develop an efficient multi-task model and mitigates the hardware overhead for resource-constrained devices. Currently, most works mainly focus on achieving MTL in deep neural networks (DNN) [2], [3], [4], [5]. However, the high complexity of DNN limits its deployability for practical applications where devices have limited resources. To overcome the limited-resource issue, the authors of [33] proposed a task offloading scheme to offload computation-intensive applications to remote edge nodes. The authors of [34] also proposed on-device multi-task transfer learning to tackle data scarcity on edge devices. Nevertheless, varying network conditions and limited computation resources of edge nodes cause complex offloading decisions and resource allocation problems [35]. Therefore, it is highly desired to devise an alternative computing scheme that can efficiently handle multiple cognitive tasks for IoT applications on devices.

Brain-inspired hyperdimensional computing (HDC) emulates the patterns of neural activities in brains and handles cognitive tasks with high-dimensional (e.g., thousands) vectors, called hypervectors (HV’s) [8]. With a simple set of well-defined vector operations, HDC exhibits many technical merits, including high energy efficiency [7], [11], robustness against hardware failures [6], and fast/few-shot learning capability [9], [10]. Moreover, HDC has shown great success in various real-world tasks, such as emotion recognition [12], seizure detection [13], DNA sequencing [14], speech recognition [15], human activity recognition [16], and image classification [17]. In some physiological applications, HDC even surpasses the state-of-the-art classification methods [9], [10], [38]. These properties and advantages both make HDC suitable for efficient inference on edge devices. For application of HDC to MTL, the authors of [32] utilized the concept of HDC to superimpose many subject-specific trained convolutional neural networks (CNNs) into one single model, reducing the memory requirement for storing CNN parameters. However,
the computation of CNN is still unaffordable for many edge devices.

In HDC, training samples based on their corresponding classes are transformed and merged into a set of representative HVs, called class HVs. Class HVs are stored in the associative memory (AM) and are used to determine the predicted label in the inference phase.

Despite the superior computational efficiency of HDC, under the scenario of multi-task learning-based HDC (MTL-HDC), the overall size of HDC model considerably grows with the number of tasks since HDC stores a separate AM for each task. As the number of tasks increases, the huge memory overhead of AMs becomes a significant problem. Although AMs of different tasks can be directly merged into one shared by all tasks to reduce memory overhead; however, this approach causes unacceptable performance degradation due to severe interference between AMs. Namely, MTL-HDC confronts a trade-off between memory overhead and performance degradation, as illustrated in Fig. 1.

To mitigate the interference while reducing the size of AMs, TP-HDC [18] randomly projects AMs into separate subspaces for learning. Meanwhile, IP-HDC [19] exploits redundant dimensionality in the high-dimensional (HD) space to extract informative dimensions of each task and merge them. Both of the prior works demonstrate the feasibility of applying MTL to HDC. However, there are still some open issues:

1) The trade-off between information and interference should be further improved: In IP-HDC, informative dimensions of each task are extracted and merged into a shared AM, as shown in Fig. 2a. Specifically, a reservation ratio determines the proportion of dimensions reserved for each task. However, the reservation ratio ignores the trade-off between the reserved information for each task and interference among them. This issue causes performance degradation of the shared AM.

2) Existing model compression-based approaches to MTL-HDC ignore label information: Both TP-HDC and IP-HDC merge multiple AMs into a shared AM. Each dimension contains information from multiple tasks. SparseHD [20] observes that dropping some dimensions in the HD space does not pose a significant impact on accuracy. Intuitively, this indicates that we may establish an independent AM with a smaller size for each task, as shown in Fig. 2b. While SparseHD demonstrates a dramatic improvement in memory efficiency for many real-world tasks, we argue that this work ignores label information in the training dataset, which may lead to a suboptimal solution.

This paper aims to establish an MTL-HDC framework that achieves a flexible and efficient trade-off between memory overhead and performance degradation. Our main contributions are summarized as follows:

1) For a shared-AM-based MTL-HDC system, we propose Dimension Ranking for Effective AM Sharing (DREAMS) to merge multiple AMs while striking the best balance between information reserved for each task and interference among them. To rank each dimension’s importance, we propose supervised dimension ranking (SDR) to consider label information in the training set. Moreover, different from IP-HDC, which solely uses the reservation ratio, we introduce an additional degree of freedom to DREAMS and improve its flexibility. Compared with TP-HDC and IP-HDC, DREAMS shows better accuracy of 11.74% on average.

2) For an independent-AM-based MTL-HDC system, we propose Dimension Ranking for Independent Memory Retrieval (DRIMER) to store small-sized AM for each task independently. Moreover, we investigate the pros and cons of the shared AM and the independent AMs under two kinds of application scenarios. When the storage space on edge devices is large enough, independent AMs perform better than the shared AM since there is no interference between tasks. On the contrary, with insufficient storage space, the shared AM can reserve more discriminative information for each task and achieve higher performance.

3) By leveraging the combined strengths of the shared-AM and the independent-AM approaches, we propose a hybrid framework of Multi-Tasking HDC, called MulTa-HDC. Depending on the memory capacity of edge devices, MulTa-HDC can flexibly adjust the proportion of the shared AM and independent AMs by parameter tuning. To evaluate the effectiveness of MulTa-HDC, we extensively experiment on three benchmark datasets, including ISOLET [21], UCIHAR [22], and MNIST [23], under two
task protocols for MTL [26]. Simulation results show that MulTa-HDC outperforms prior works and achieves an efficient trade-off between memory overhead and performance degradation.

The rest of the paper is organized as follows. In Section 2, we review HDC algorithm and describe existing approaches that handle the problem of MTL-HDC. Section 3 illustrates the proposed hybrid framework of MulTa-HDC. The proposed DREAMS and DRIMER are presented in Section 4. Section 5 presents simulation results and evaluations. Finally, we conclude this paper in Section 6.

2 Preliminaries

2.1 HDC Algorithm

HDC operates on high-dimensional HVs, whose elements are with equally probable (-1)s and 1s. Assume there are \( m \) randomly selected bits of \( L_1 \) are flipped to generate an HV for the next level, and so on. The bit-flipping approach ensures a high correlation between adjacent levels. Therefore, the procedure for mapping the \( i \)th feature value of \( f_i \) comprises discretization and looking up the corresponding vector \( S_i \) from the CiM. After mapping each feature component of \( F \), a set of two-vector pairs \( P = \{(ID_1, S_1), (ID_2, S_2), \ldots, (ID_m, S_m)\} \) is obtained.

2.1.2 HD Encoding

In HD encoding, HDC utilizes two dimension-wise vector operations to encode the pairs in \( P \) to a single representative vector \( H \in \{-1, 1\}^D \):

\[
H = \langle \sum_{i=1}^m S_i \oplus ID_i \rangle, \tag{1}
\]

where \( \oplus \) is defined as a binding operation to perform dimension-wise XOR between two HVs, \( + \) is defined as a bundling operation to compute dimension-wise addition, and \( \langle \cdot \rangle \) is defined as the majority function to bipolarize each component of HVs with the threshold 0.

2.1.3 Training Phase

In the training phase, all the same class’s encoded training data are summed up to form a class HV. Assume that the \( j \)th class has \( n_j \) samples associated with a set of encoded HVs \( \{H_1, H_2, \ldots, H_{n_j}\} \), a class HV \( AM_j \in \mathbb{R}^D \) is computed as:

\[
AM_j = \sum_{i=1}^{n_j} H_i. \tag{2}
\]

For a \( C \)-class classification task, \( C \) class HVs denoted as \( \{AM_1, AM_2, \ldots, AM_C\} \) are generated and stored in the AM.

2.1.4 Inference Phase

In the inference phase, a testing sample is encoded as a query vector \( Q \in \{-1, 1\}^D \) by (1). Next, HDC evaluates the similarity between \( Q \) and all class HVs stored in AM. In this work, to achieve efficient similarity checking, class HVs are bipolarized to simplify the similarity metric as Hamming distance computation. The class with the highest similarity is the prediction:

\[
\text{Prediction} = \arg \max_{j} (\text{Hamming}(Q, AM_j)). \tag{3}
\]

2.2 Multi-task Learning-Based HDC (MTL-HDC)

2.2.1 Problem Description of MTL-HDC

Assume there are \( T \) tasks to be solved in the MTL-HDC system, and each of them contains \( C \) classes. As shown in Fig. 4, an AM table stores \( T \) isolated AMs. \( AM_{i,j} \) represents the \( j \)th class HV of the \( i \)th task. However, this approach suffers from huge memory overhead when \( T \) increases. On the other hand, one can directly bundle all of the \( j \)th class HVs from \( T \) tasks \( \{AM_{1,j}, AM_{2,j}, \ldots, AM_{T,j}\} \) to get \( AM_{j} \), which is shared across tasks. However, the performance
dramatically degrades due to interference between tasks. Hence, the problem of MTL-HDC is how to strike a balance between memory storage and performance degradation while transforming the AM table into a single AM whose size is independent of the number of tasks $T$.

### 2.2.2 Shared-AM Approach to MTL-HDC

The shared-AM approach manages to merge multiple AMs from different tasks into a shared one, as shown in Fig. 5a. To achieve AM sharing, TP-HDC [18] and IP-HDC [19] have been proposed to reduce the interference while merging AMs. TP-HDC exploits the model capacity by randomly projecting AMs into mutually orthogonal HD subspaces for merging. Meanwhile, IP-HDC analytically performs dimension ranking to filter out relatively inconsequential dimensions before AM sharing, leading to interference mitigation between tasks.

### 2.2.3 Independent-AM Approach to MTL-HDC

The independent-AM approach mitigates AM table's memory overhead by model compression, as shown in Fig. 5b. It reduces the required number of dimensions in each AM. With fewer dimensions brought by each AM, the total size of the compressed AMs is equal to that of a single uncompressed AM. Furthermore, no interference between tasks occurs due to isolation between the compressed AMs. Two prior works realize model compression for the independent-AM approach, including CompHD [25] and SparseHD [20]. CompHD compresses an HV into shorter segments and leverages orthogonality to decompose them with noise mitigated. SparseHD analytically measures variation in the class HVs for each dimension and discards inconsequential dimensions without significant impacts on classification results.

### 2.2.4 Comparisons of the Two Approaches

The shared-AM and independent-AM approaches utilize different properties of HDC to realize an MTL-HDC system, and we summarize them in Table 1. These two approaches are intrinsically the same, which are related to exploiting the model capacity of HDC efficiently. However, for the shared-AM approach, prior works [18] [19] do not fully consider the balance between the reserved information for each task and interference among them. For the independent-AM approach, the compression methods can be further improved by leveraging the label information in the training set. Moreover, we observe that the two approaches have different and complementary behaviors under two application scenarios, corresponding to different memory resources on edge devices. This means that neither approach is complete without the other. Therefore, a framework for MTL-HDC that can handle both issues is desirable. In the next section, we first analyze the strengths and weaknesses of shared-AM and independent-AM approaches. Next, we propose a hybrid framework for MTL-HDC based on both approaches to achieve efficient memory scalability, as shown in Fig. 5c.

## 3 Multi-Task Learning Framework for HDC

### 3.1 Observation of Shared and Independent AMs

We experiment with both approaches on the Permuted UCI-HAR dataset, which is explained in Section 5.1. Five tasks are in the MTL-HDC system with HD dimensionality $D$ ranging from 1K to 9K. Different HD dimensionalities imply various amounts of available memory capacity, corresponding to different application scenarios. Fig. 6 presents the classification accuracy of MTL-HDC based on shared-AM, independent-AM, and hybrid approaches. The independent-AM approach realized by our proposed DRIMER, which is discussed in Section 4.3, has stable performance in high HD dimensionality. Therefore, when designing an MTL system on edge devices with large memory capacity, the independent-AM approach is better than the shared one since there is no interference between independent AMs.

However, we observe a critical point where a smaller HD dimensionality leads to a significant accuracy drop for the independent-AM approach. Specifically, when $D < 4000$,
the compressed AMs can only retain an insufficient amount of discriminative information for classification. When devices only allow a maximum HD dimensionality lower than the critical point, the independent-AM approach is not reliable. On the contrary, the shared-AM approach based on our proposed DREAMS, which is discussed in Section 4.2, can mitigate the accuracy drop. The shared AM accommodates more information at the costs of interference between tasks. The positive effect brought by more reserved information is larger than the negative impact caused by interference.

Based on the investigation, these two approaches have complementary behaviors under different application scenarios, as shown in Table 2. The independent-AM approach is more suitable for devices with large memory capacity since the independent AMs introduce no interference between tasks. The shared-AM approach is better for memory-constrained devices since they can reserve more discriminative information for each task, even if interference between tasks exists.

### Table 1: Comparisons of Existing Approaches to MTL-HDC

<table>
<thead>
<tr>
<th>Frameworks</th>
<th>Analytic Projection</th>
<th>Consider Label Information</th>
<th>Information/Interference Balancing</th>
<th>Memory Scalability</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>TP-HDC [18]</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>Orthogonality</td>
</tr>
<tr>
<td>IP-HDC [19]</td>
<td>◊</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>Interference Mitigation</td>
</tr>
<tr>
<td>Proposed DREAMS</td>
<td>◊</td>
<td>◊</td>
<td>◊</td>
<td>×</td>
<td>Interference Mitigation</td>
</tr>
<tr>
<td>CompHD [25]</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>×</td>
<td>Orthogonality</td>
</tr>
<tr>
<td>SparseHD [20]</td>
<td>◊</td>
<td>×</td>
<td>–</td>
<td>×</td>
<td>Dimension Redundancy</td>
</tr>
<tr>
<td>Proposed DRIMER</td>
<td>◊</td>
<td>◊</td>
<td>◊</td>
<td>×</td>
<td>Dimension Redundancy</td>
</tr>
<tr>
<td>Proposed Multi-HDC</td>
<td>◊</td>
<td>◊</td>
<td>◊</td>
<td>◊</td>
<td>Dimension Redundancy &amp; Interference Mitigation</td>
</tr>
</tbody>
</table>

### Table 2: Comparisons of Shared-AM and Independent-AM Approaches to MTL-HDC

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Low HD Dimensionality</th>
<th>High HD Dimensionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared AM</td>
<td>More reserved information</td>
<td>Severe interference</td>
</tr>
<tr>
<td>Independent AM</td>
<td>Insufficient retained information</td>
<td>Stable due to no interference</td>
</tr>
</tbody>
</table>

### 3.2 Proposed Hybrid Framework – Multi-HDC

We propose Multi-HDC to integrate two approaches for MTL-HDC into one hybrid framework to exploit their strengths under different application scenarios. Moreover, DREAMS and DRIMER are proposed to improve the performance of the shared-AM and independent-AM approaches, respectively. First, we introduce three parameters in the hybrid framework. We define the independent ratio \( \alpha \) as:

\[
\alpha = \frac{D^{\text{ind}}}{D},
\]

where \( D^{\text{ind}} \) is the dimensionality assigned to independent AMs. In other words, each task in the system can store and retrieve its independent AM with \( D^{\text{ind}}/T \) dimensionality. The remaining \( D^{\text{sha}} \) dimensions in the system are assigned to the shared AMs:

\[
D^{\text{sha}} = (1 - \alpha)D.
\]

The proposed DRIMER takes \( \alpha \) as input and generates independent AMs for each task. For the remaining \( D(1 - \alpha) \) dimensions, the proposed DREAMS takes the screening ratio \( \beta \) and the reservation ratio \( \gamma \) as inputs to merge...
TABLE 3
Notations for MulTa-HDC

<table>
<thead>
<tr>
<th>Associative Memory</th>
<th>Description</th>
<th>Store</th>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AM_{ij}$</td>
<td>Single AM</td>
<td>AM</td>
<td>$AM_{ij} \in \mathbb{R}^D$</td>
</tr>
<tr>
<td>$AM_{Uj}$</td>
<td>AM table</td>
<td></td>
<td>$AM_{Uj} \in \mathbb{R}^D$</td>
</tr>
<tr>
<td>$AM_{ij}^{str}$</td>
<td>Screened AMs in DREAMS</td>
<td>AM</td>
<td>$AM_{ij}^{str} \in \mathbb{R}^{D_B}$</td>
</tr>
<tr>
<td>$AM_{ij}^{mer}$</td>
<td>Merged AM in DREAMS</td>
<td>AM</td>
<td>$AM_{ij}^{mer} \in \mathbb{R}^{D_B}$</td>
</tr>
<tr>
<td>$AM_{sha}$</td>
<td>Shared AM of DREAMS</td>
<td>AM</td>
<td>$AM_{sha} \in \mathbb{R}^{D_B}$</td>
</tr>
<tr>
<td>$AM_{sha}^{dec}$</td>
<td>Decoded AMs of DREAMS</td>
<td>AM</td>
<td>$AM_{sha}^{dec} \in \mathbb{R}^{D_B}$</td>
</tr>
<tr>
<td>$AM_{ind}$</td>
<td>Independent AMs of DRIMER</td>
<td>AM</td>
<td>$AM_{ind} \in \mathbb{R}^{D_B}$</td>
</tr>
<tr>
<td>$K_{i}^{scr}$</td>
<td>Screen dimensions of $AM_i$ to form $AM_i^{str}$</td>
<td></td>
<td>$K_{i}^{scr} \in (0,1)^D$, $|K_{i}^{scr}|_0 = D_B$</td>
</tr>
<tr>
<td>$K_{i}^{mer}$</td>
<td>Filter dimensions of $AM_i^{str}$ to merge into $AM_i^{mer}$</td>
<td></td>
<td>$K_{i}^{mer} \in (0,1)^D_B$, $|K_{i}^{mer}|_0 = D_B\gamma$</td>
</tr>
<tr>
<td>$K_{i}^{rd}$</td>
<td>Indicate the shared dimensions of $AM_i^{mer}$</td>
<td></td>
<td>$K_{i}^{rd} \in (0,1)^D_B$, $|K_{i}^{rd}|_0 = D_B\gamma$</td>
</tr>
<tr>
<td>$K_{i}^{dec}$</td>
<td>Extract dimensions of $AM_i^{sha}$ to form $AM_i^{dec}$</td>
<td>$\circledast$</td>
<td>$K_{i}^{dec} \in (0,1)^{D_B\gamma}$, $|K_{i}^{dec}|_0 = D_B\gamma$</td>
</tr>
<tr>
<td>$K_{i}^{sha}$</td>
<td>Extract dimensions of $Q_i$ to check with $AM_i^{dec}$</td>
<td>$\circledast$</td>
<td>$K_{i}^{sha} \in (0,1)^D$, $|K_{i}^{sha}|_0 = D_B\gamma$</td>
</tr>
<tr>
<td>$K_{i}^{ind}$</td>
<td>Extract dimensions of $Q_i$ to check with $AM_i^{ind}$</td>
<td>$\circledast$</td>
<td>$K_{i}^{ind} \in (0,1)^D$, $|K_{i}^{ind}|_0 = D_B\gamma$</td>
</tr>
</tbody>
</table>

$AM$s from different tasks into a shared one. To sum up, the optimization problem of MulTa-HDC can be formulated as:

\[
\max_{\alpha, \beta, \gamma} \left\{ \frac{1}{T} \sum_{t=1}^{T} Acc_t(AM_i^{ind}, AM_i^{sha}) \right\},
\]

\[s.t. \quad \alpha D_B + D_B\gamma (\beta, \gamma, AM) = D_B,
\]

where $AM$ denotes the AM table. $D_B\gamma$ is $AM$-dependent and related to both ratios $\beta$ and $\gamma$. $Acc_t(AM_i^{ind}, AM_i^{sha})$ is the classification accuracy of the $t$th task based on the independent $AM_i^{ind}$ and the shared $AM_i^{sha}$.

Throughout this paper, there are various kinds of AMs in MulTa-HDC. We summarize the notations for them in Table 3. A normal-faced and italic type $AM$ denotes an ordinary $AM$, where $AM_{ij}$ denotes the $j$th class HV in $AM$. A bold-faced and italic type $AM$ denotes an AM table, where $AM_{ij}$ denotes the $j$th class HV of the $i$th task in $AM$. Besides, mask HVs are binary vectors necessary for MulTa-HDC, which are introduced in Section 4. The second column of Table 3 implies whether an instance should be stored in devices for on-line inference.

Fig. 7 shows how MulTa-HDC transforms an AM table into a hybrid AM. The processing flow is summarized in Algorithm 1. In the off-line phase, each task in the MTL-HDC system evaluates each dimension’s importance in the HD space by the proposed SDR, which is introduced in Section 4.1. Based on SDR, DREAMS and DRIMER extract two disjoint sets of dimensions from the AM table and generate the shared AM $AM_i^{sha}$ and independent AMs $\{AM_i^{ind} \text{ for } i = 1, 2, \ldots, T\}$, respectively.

**Algorithm 1. MulTa-HDC Framework**

**Input:** AM table $AM_i$, $\alpha$, $\beta$, $\gamma$  
**Output:** $\{AM_i^{ind} \text{ for } i = 1, 2, \ldots, T\}$, $AM_i^{sha}$, $K_i^{sha}$, $K_i^{dec}$, $K_i^{ind}$

**On-line phase for the query vector $Q_i$ of the $i$th task:**

1. $AM_i^{ind} \text{, } K_i^{ind} = \text{DRIMER.Train( } AM_i, \alpha \text{)}$
2. $AM_i^{sha} \text{, } K_i^{sha} \text{, } K_i^{dec} = \text{DREAMS.Train( } AM_i, \alpha, \beta, \gamma \text{)}$
3. $s_i^{sha} = \text{DRIMER.Test( } Q_i, AM_i^{sha}, K_i^{sha} \text{)}$
4. $s_i^{ind} = \text{DREAMS.Test( } Q_i, AM_i^{sha}, K_i^{sha}, K_i^{dec} \text{)}$
5. Prediction = $\arg \max_j (s_i^{sha}[j] + s_i^{ind}[j])$

In the on-line inference phase, an unseen query vector $Q_i$ of the $i$th task comes in. MulTa-HDC checks the similarity between $Q_i$ and the hybrid AM to obtain similarity scores $s_i^{sha}$, $s_i^{ind}$ from the independent AMs and the shared AM, respectively. Finally, the class with the highest accumulated similarity score is the prediction.

Compared with the case using either the shared-AM or the independent-AM approach, MulTa-HDC can flexibly adjust the proportion of two kinds of AMs to strike the right balance between memory storage and performance degradation. Note that the shared-AM and the independent-AM approach can be viewed as the subsets of MulTa-HDC when $\gamma$ is set as zero and one, respectively. In the following section, we present how to effectively generate the shared AM and the independent AMs by the proposed DREAMS and DRIMER, respectively.

**4 PROPOSED DREAMS AND DRIMER**

In this section, we introduce DREAMS and DRIMER. DREAMS comprises two techniques: 1) Dimension Ranking, which evaluates the importance of each dimension, and 2) AM Sharing, which merges AMs from different tasks into a shared one. DRIMER also leverages two techniques: 1) Dimension Ranking, which performs the same functionality as in DREAMS, and 2) AM Retrieval, which independently stores AM of each task for retrieval. We first present how to rank dimensions while considering label information, followed by the details of DREAMS and DRIMER.
4.1 Supervised Dimension Ranking (SDR)

SDR ranks the importance of each dimension in the HD space by leveraging label information, as shown in Fig. 8a and summarized in Algorithm 2. For the \(i^{th}\) task in the MTL-HDC system, SDR takes as input \(AM_i\) in non-binary representation and all encoded HVs from the training dataset, denoted \(H^i = \{H_1, H_2, \ldots, H_N\}\). To evaluate the importance of each dimension in a supervised manner, we define a score \(\tau \in \mathbb{R}^D\) for each training data as:

\[
\tau = H \odot AM_{i,p} - \sum_{j \neq p} H \odot AM_{i,j}, \tag{7}
\]

where \(H\) is an encoded HV of a training sample belonging to the \(i^{th}\) class. The \(\odot\) operation denotes element-wise multiplication between two HVs. From the viewpoint of a specific dimension \(d\), variation between \(AM_{i,d}\) implies the change in class elements, which can be viewed as the statistics of AM. On the other hand, \(H[d]\) can be regarded as a scaling factor that decides the influence of the \(d^{th}\) dimension. Therefore, larger values in the score HV indicate more discriminative information useful for the HD model to correctly classify \(H\). After aggregating \(N\) score HVs, an information HV \(I \in \mathbb{R}^D\) is obtained as:

\[
I = \sum_{n=1}^{N} \tau_n, \tag{8}
\]

**Algorithm 2. Supervised Dimension Ranking (SDR)**

**Input:** AM of the \(i^{th}\) task \(AM_i\).

Encoded HVs from the training set of the \(i^{th}\) task \(H^i = \{H_1, H_2, \ldots, H_N\}\)

**Output:** Information HV \(I\)

1: Initialize \(I \leftarrow 0\)
2: for \((n = 1:N)\) do
3:    Initialize \(\tau_n = 0\)
4:    for \((j = 1:C)\) do
5:        if \(H_n\) belongs to the \(j^{th}\) class
6:            \(\tau_n = \tau_n + H_n \odot AM_{i,j}\)
7:        else
8:            \(\tau_n = \tau_n - H_n \odot AM_{i,j}\)
9:        end if
10:    end for
11:  \(I = I + \tau_n\)
12: end for

where \(\tau_n\) denotes the score HV derived from the \(n^{th}\) training data. The information HV reveals the importance of each dimension in the HD space over the whole training dataset. Fig. 8b shows the histogram of the distribution of values in \(I\). We observe that each dimension contains imbalanced amounts of information, implying a chance to extract important information from the given task systematically. Therefore, those extracted dimensions can be further exploited to obtain a shared AM or independent AMs by DREAMS and DRIMER, respectively. The details are elaborated in Section 4.2 and 4.3, respectively.

4.2 DREAMS for Shared AM

The goal of AM sharing is to alleviate the huge memory overhead brought by the AM table. However, interference between tasks in the shared AM leads to potential performance degradation. Hence, we propose DREAMS to generate a shared AM while striking the right balance between the reserved information for each task and interference among them, as shown in Fig. 9 and summarized in Algorithm 3. In the off-line phase, we propose a two-stage AM sharing mechanism by utilizing the screening ratio \(\beta\) and the reservation ratio \(\gamma\) to improve the utilization of the model capacity of HDC. In the on-line inference phase, the task-specific AM from the shared AM is decoded for classification.

4.2.1 Two-Stage AM Sharing System

In the off-line phase, DREAMS first evaluates the importance of each dimension by SDR and generates the information HV \(I_i\) for \(i = 1, 2, \ldots, T\). Rather than directly merging multiple AMs into a shared one with severe interference among tasks, we retain the best representative dimensions with greater values in \(I_i\) for each task. Those dimensions with smaller values in \(I_i\) are considered as noise and thus should be screened out. To achieve this, we introduce several mask HVs generated by Algorithm 4 (MG) to indicate which dimensions are reserved in the shared AM and activated in the on-line phase. Based on the screening ratio \(\beta\), reservation ratio \(\gamma\), and information HVs \(I_i\), DREAMS
decomposes the AM merging process into two stages: 1) AM screening, and 2) AM sharing.

1) AM screening. In the AM screening stage, the screening mask HVs, denoted as \(K_{scr}^i \) for \( i = 1, 2, \ldots, T \), are obtained based on the information HV \( I_i \), the independent ratio \( \alpha \), and the screening ratio \( \beta \), as shown in Fig. 9a. If \( \alpha \neq 0 \), then a total of \( D \times aT \) dimensions in \( AM_i \) are retained in the independent AM for the \( i \)th task. Meanwhile, \( K_{scr}^i \) indicates the most significant \( \beta D \) dimensions in the remaining part of \( AM_i \). The screening ratio \( \beta \) determines the number of dimensions that are retained for the next stage. In other words, we only extract and concatenate \( \beta D \) important dimensions by Algorithm 5 (EAC) and discard others. The screened AMs \( AM_{scr}^i = \{ K_{scr}^i \} \) for \( i = 1, 2, \ldots, T \) and the screened information HVs \( l_{scr}^i \) for \( i = 1, 2, \ldots, T \) are obtained based on \( K_{scr}^i \) and are transferred to the next stage.

Algorithm 3. Dimension Ranking for Efficient AM Sharing (DREAMS)

**Input:** AM table \( AM, \alpha, \beta, \gamma, T \)

- Encoded HVs from the training set of each task \( \{ \vec{h} \} \) for \( i = 1, 2, \ldots, T \)

**Output:** \( \{ K_{scr}^i \} \) for \( i = 1, 2, \ldots, T \), \( AM_{sha} \), \( \{ K_{dec}^i \} \) for \( i = 1, 2, \ldots, T \), \( D_{sha} \) = \( \| K_{RD} \|_0 \)

1. Initialize \( K_{RD} \) = \( 0 \) \( \beta D \); \( AM_{mer} \) = \( 0 \) \( \beta D \)

Off-line phase:

2. for \( (i = 1:T) \) do
3: \( I_i = SDR(AM_i, H^i) \)
4: \( K_{scr}^i = MG(I_i, aT, aT + \beta) \)
5: \( AM_{scr}^i = EAC(K_{scr}^i, AM_i) \)
6: \( l_{scr}^i = EAC(K_{scr}^i, l_i) \)
7: \( K_{mer}^i = MG(l_{scr}^i, 0, \gamma) \)
8: \( AM_{mer} = AM_{mer} + AM_{scr}^i \odot K_{mer}^i \)
9: for \( (d = 1: \beta D) \) do
10: if \( K_{mer}^i[d] \neq 0 \)
11: \( K_{RD}[d] = 0 \)
12: end if
13: end for
14: end for
15: \( AM_{sha} = EAC(K_{RD}, AM_{mer}) \)
16: for \( (i = 1:T) \) do
17: \( K_{dec}^i = EAC(K_{RD}, K_{mer}^i) \)
18: \( K_{sha}^i = MG(I_i, 0, \beta \gamma) \)
19: end for
20: On-line phase for the query HV \( Q_i \) of the \( i \)th task:
21: \( Q_{sha}^i = EAC(K_{sha}^i, Q_i) \)
22: \( AM_{dec}^i = EAC(K_{dec}^i, AM_{sha}) \)
23: Similarity Checking (\( Q_{sha}^i, AM_{dec}^i \))

2) AM sharing. Fig. 9b illustrates the AM sharing stage. DREAMS uses the reservation ratio \( \gamma \) to determine the number of important dimensions that are retained in the merged AM \( AM_{mer} \). The merging mask HVs, denoted as \( K_{mer}^i \) for \( i = 1, 2, \ldots, T \), are obtained based on \( \gamma \) and the screened information HVs from the previous stage \( l_{scr}^i \) for \( i = 1, 2, \ldots, T \). They help merge the screened AMs \( \{ AM_{mer}^i \} \) for \( i = 1, 2, \ldots, T \) into \( AM_{mer} \):

\[
AM_{mer}^i = \sum_{i=1}^{T} AM_{mer}^i \odot K_{mer}^i , \tag{9}
\]

where every row of \( A \odot B \) corresponds to the element-wise multiplication of the vector B and the corresponding row in the matrix A. In the merging process described by (9), the merging mask HV \( K_{mer}^i \) only activates the \( \beta D \) dimensions in the \( i \)th screened AM \( AM_{mer}^i \) if \( K_{mer}^i[d] = 1 \). This mitigates the interference between tasks while merging the screened AMs. Furthermore, \( \gamma \) imposes sparsity over dimensions in the merged AM \( AM_{mer} \). We define a redundancy mask HV \( K_{RD} \in (0, 1)^D \) as:

\[
K_{RD}[d] = \begin{cases} 0, & \sum_{i=1}^{T} K_{mer}^i[d] = 0, \\ 1, & \text{Otherwise} \end{cases} , \tag{10}
\]

which records the indices where none of the tasks in the MTL-HDC system stores information in \( AM_{mer} \). Since the \( \beta D \) dimension of \( AM_{mer} \), where \( K_{RD}[d] = 0 \), is always inactivated in the inference phase, it can be removed from \( AM_{mer} \) by Algorithm 5 (EAC). A more compact shared AM \( AM_{sha} \) with less memory overhead is obtained. Note that \( D_{sha} = \| K_{RD} \|_0 \) is exactly the number of dimensions assigned to the shared AM \( AM_{sha} \) in MulTa-HDC.

4.2.2 AM Decoding for Inference

In the off-line phase, a set of \( \{ K_{dec}^i \} \) for \( i = 1, 2, \ldots, T \) is already generated based on \( K_{RD} \) and \( \{ K_{mer}^i \} \) for \( i = 1, 2, \ldots, T \), as shown in Fig. 9c. Assume that the MTL-HDC system is aware of which task should be inferenced at runtime [36], [37]. In the on-line inference phase, since multiple AMs are merged into a shared one \( AM_{sha} \), DREAMS uses \( K_{dec}^i \) to decode the task-specific AM of the \( i \)th task \( AM_{dec}^i \), as shown in Fig. 9e.

Algorithm 4. Mask Generator (MG)

**Input:** Information HV \( I_i, r_{start}, r_{end} \)

**Output:** Mask HV \( K \)

1: Initialize \( D \) length(i); \( K = 0 \)
2: Sort \( I \) in ascending order to obtain \( I_{sort} \)
3: Derive start threshold value \( e_{start} \)
4: Derive end threshold value \( e_{end} \)
5: for \( (d = 1: D) \) do
6: if \( e_{start} \geq I[d] \geq e_{end} \)
7: \( K[d] = 1 \)
8: end if
9: end for

As for the query HV \( Q_i \) of the \( i \)th task, \( K_{sha}^i \) is also generated in the off-line phase to activate the reserved dimensions in \( Q_i \), as shown in Fig. 9d. The activated and concatenated dimensions \( Q_{sha}^i \) is obtained by Algorithm 5 (EAC). Note that \( \| K_{sha}^i \|_0 = D \beta \gamma \), meaning that a total of \( D \beta D \) dimensions from the \( i \)th task are reserved in the shared AM \( AM_{sha} \) and decoded as \( AM_{dec}^i \). With \( Q_{sha}^i \) and \( AM_{dec}^i \), DREAMS evaluates the similarity values \( s_{sha}^i \in R^C \) between \( Q_{sha}^i \) and the decoded class HVs stored in \( AM_{dec}^i \):

\[
s_{sha}^i[j] = \sum_{d=1}^{D \beta \gamma} Q_{sha}^i[d] < AM_{dec}^{i,j} > \quad \text{for} \quad j = 1, \ldots, C. \tag{11}
\]
Algorithm 5. Extract And Concatenate (EAC)

**Input:** Mask HV $K$, AM or HV $Q$

**Output:** $AM^{EAC}$ or $Q^{EAC}$

1. Initialize $AM^{EAC} = 0^{|K|_D}$; $Q^{EAC} = 0^{|K|_D}$; count = 1
2. if input is $AM$
3. for ($j = 1$; $C$) do
4. $AM_j^{EAC} = EAC(K, AM_j)$
5. end for
6. else
7. for ($d = 1$; length($K$)) do
8. if $K[d] 
eq 0$
9. $Q^{EAC}[\text{count}] = Q[d]$
10. count = count + 1
11. end for
12. end if

4.2.3 The Flexibility of the Two-Stage AM Sharing

The two-stage AM sharing mechanism provides two degrees of freedom to improve the flexibility of DREAMS. We observe that $D^{sha} = ||K^{RD}||_0$ of DREAMS depends on the $(\beta, \gamma)$ pair, but not $(\beta \times \gamma)$ only. Specifically, different combinations of $(\beta, \gamma)$ pairs may result in the same $D^{sha}$, but with different numbers of reserved dimensions for each task, which is $D^{c}$. Compared with the merging method in IP-HDC [19] as the particular case of DREAMS with $\beta = 1$, the flexibility brought by the two-stage mechanism enable DREAMS to achieve a better balance between interference and information by searching the optimal $(\beta, \gamma)$ pair. We validate the effectiveness of DREAMS in Section 5.2.

4.2.4 MTL Scenario With Varied Number of Classes

In a real-world MTL-HDC system, AM’s size in each task depends on the number of classes it contains. DREAMS retains important dimensions in each AM and merge them into one. However, the merging procedure corresponding to Line 8 of Algorithm 3 is not feasible when AM’s size differs. Specifically, the screened AM in DREAMS has the size of $(C_i \times D^\beta)$, where $C_i$ and $\beta$ denote the number of classes of the $i^{th}$ task and the screening ratio, respectively. When $C_i \neq C_j$ for any $i \neq j$, the screened AM cannot be merged directly.

Suppose that in an MTL-HDC system, the $i^{th}$ task does not contain the $j^{th}$ class (i.e., $AM_{i,j}^{scr}$ does not exist). As an intuitive approach, a pseudo class HV which corresponds to the $j^{th}$ class of the $i^{th}$ task can be a “zero” HV, which contains zeros in all HD dimensions. However, this approach causes imbalanced noises of each class from the $i^{th}$ task, leading to severe performance degradation. The imbalanced noises are in favor of the $j^{th}$ class in other tasks, since they are not affected by the noise from the $i^{th}$ task. Instead, to simulate the noise caused by naturally generated HVs, we add pseudo classes to the $i^{th}$ task by randomly-weighted averaging the inherent class HVs in its AM. For example, in Fig. 10, the MTL system originally has two tasks, where task 1 has three classes, task 2 has only two classes. The pseudo class HV of task 2, which is denoted as $AM_{2,3}^{sha}$, is the randomly-weighted average of the other two class HVs. Note that pseudo classes are excluded from the similarity checking process in the inference phase, based on the prior knowledge on the number of classes. The method seems
4.3 DRIMER for Independent AMs

DRIMER obtains an independent and compressed AM by discarding inconsequential dimensions for each task. Different from the shared AM, the independent AMs have no interference between tasks. To distinguish from the screened AMs \( A M_{text}^{ind} \), the independently stored AMs is referred to as the independent AMs \( A M_{i}^{ind} \), for \( i = 1, 2, \ldots, T \). Fig. 11 illustrates the processing flow of DRIMER, which is summarized in Algorithm 6.

4.3.1 Independent AMs Based on SDR

To drop inconsequential dimensions in AM, DRIMER uses SDR to obtain the information HVs \( \{ I_i \) for \( i = 1, 2, \ldots, T \) \) in off-line phase. Then, the first \( aD \) dimensions with the highest score values in \( I_i \) are selected to be retained in the compressed AM. The independent mask HV \( K_i^{ind} \) is obtained based on the independent ratio \( \alpha \) and \( I_i \). As shown in Fig. 11a, \( K_i^{ind} \) decides which dimensions are retained to form the \( i^{th} \) independent AM \( A M_i^{ind} \), reducing the memory size by a factor of \((1 - \alpha)\).

In the on-line inference phase, Fig. 11b shows the retrieval of the compressed AM for the specific task. \( K_i^{ind} \) indicates the dimensions in the query HV \( Q_i \) that are activated for inference. Those activated dimensions are extracted and become \( Q_i^{ind} \). The similarity value \( s_i^{ind}[j] = \sum_{d=1}^{D_{ind}} Q_i^{ind}[d] < A M_{i,j}^{ind}[d] > \), for \( j = 1, \ldots, C \).

\[
(12)
\]

Algorithm 6. Dimension Ranking for Independent Memory Retrieval (DRIMER)

<table>
<thead>
<tr>
<th>Input: AM table ( A M_i, \alpha, T )</th>
<th>Encoded HVs from the training set of each task ( { H_i ) for ( i = 1, 2, \ldots, T ) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output: ( { A M_i^{ind} ) for ( i = 1, 2, \ldots, T } )</td>
<td>( { K_i^{ind} ) for ( i = 1, 2, \ldots, T )</td>
</tr>
</tbody>
</table>

**Off-line phase:**

1: \( \text{for} \ (i = 1:T) \) do
2: \( I_i = \text{SDR}(A M_i, H_i) \)
3: \( K_i^{ind} = \text{MG}(I_i, 0, \alpha/T) \)
4: \( A M_i^{ind} = \text{EAC}(K_i^{ind}, A M_i) \)
5: \( \text{end for} \)

**On-line phase for the query vector \( Q_i \) of the \( i^{th} \) task:**

6: \( Q_i^{ind} = \text{EAC}(K_i^{ind}, Q_i) \)
7: \( \text{Similarity Checking} (Q_i^{ind}, A M_i^{ind}) \)

By considering the label information in the training set, SDR discovers more representative dimensions than prior works. This makes DRIMER reduce information loss after compressing independent AMs. We validate the effectiveness of DRIMER in Section 5.3.

4.3.2 Retraining Independent AMs

After compression, the performance degrades due to information loss. We refer to the retraining mechanism in [31] with some modifications to compensate for the quality loss. The retraining procedure first checks the predictions of the independent AMs on the training set by (12). (i) If a query vector \( Q_i \), which belongs to the \( j^{th} \) class of the \( i^{th} \) task, is correctly classified, nothing changed. (ii) If \( Q_i \) is wrongly predicted as the \( k^{th} \) class in the independent AM \( A M_i^{ind} \), the adjustment process subtracts \( Q_i^{ind} \) from \( A M_{i,k}^{ind} \) and adds it to \( A M_{i,j}^{ind} \):

\[
\begin{align*}
A M_{i,k}^{ind} & = A M_{i,k}^{ind} - Q_i^{ind} \\
A M_{i,j}^{ind} & = A M_{i,j}^{ind} + Q_i^{ind}
\end{align*}
\]

(13)

The procedure iterates over the training set until the model converges, which is defined as the fact that the validation accuracy does not improve in the last five iterations.

5 EXPERIMENTS AND EVALUATIONS

5.1 Experimental Setup

This section evaluates the performance of proposed DREAMs, DRIMER, and Multi-HDC by using three datasets, including image classification, voice recognition, and activity recognition tasks. "Permuted MNIST" [23] is a standard task protocol for MTL [26] [28] [29]. Permuted means that the input representation is randomly permuted for each task, but all MTL system tasks are standard (10-class) MNIST classification tasks. Permuted ISOLET [21] and Permuted UCIHAR [22] share the same concept as Permuted MNIST. In ISOLET, 150 subjects speak the name of 26 letters of the alphabets, and 617 features of voice signals are extracted. UCIHAR recognizes six human activities based on 3-axial linear acceleration and angular velocity at a constant rate of 50 Hz.

For hyperparameter settings, the HD dimension \( D \) and quantization level \( i \) is set as 10K and 16, respectively, where the performances of all tasks saturate. Mapping and encoding modules are shared across tasks. Besides, accommodating multiple tasks in the MTL-HDC system leads to an accuracy drop. Therefore, all models are retrained for 50 epoches.

![Fig. 11. The processing flow of DRIMER. (a) Generate the independent AMs \( A M_i^{ind} \) in the off-line phase, corresponding to lines 1-5 in Algorithm 6. (b) Retrieve task-specific AM for inference, corresponding to lines 6-7 in Algorithm 6.](image-url)
iterations before evaluation. The evaluation metric used in this paper is the averaged classification accuracy defined as:

$$\text{Accuracy} = \frac{1}{T} \sum_{i=1}^{T} \text{Acc}(AM_i^{\text{ind}}, AM^{\text{sha}}).$$ (14)

The hyperparameter settings for HDC and evaluated datasets are summarized in Table 4.

### 5.2 Evaluation of DREAMS

The proposed DREAMS strikes a balance between information reserved for each task and interference in the shared AM, improving the utilization of the model capacity. To validate the effectiveness of DREAMS, we experiment it on the Permuted ISOLET dataset. The number of tasks and classes are set as $T = 5$ and $C = 26$, respectively.

First, the utilization of the model capacity is investigated. We fix the shared dimensions $D^{\text{sha}}$ at 7K. Fig. 12 shows the results, where the horizontal axis is the screening ratio $\beta$. Specifically, for each $\beta$ value in Fig. 12, we exhaustively search for a reservation ratio $\gamma$ such that the specified $(\beta, \gamma)$ pair result in $D^{\text{sha}}$ being 7K with small variances. To quantize the amount of reserved information from each task and interference, we split up the dot product in (11) into two parts, including signal and noise. From the viewpoint of a query vector $Q_i$ of the $i^{th}$ task, signal and noise are defined as:

$$\text{Signal} = \max_j \left\{ EAC(K_i^{\text{sha}}, Q_i) \cdot EAC(K_i^{\text{mer}}), AM_{ij}^{\text{scr}} \right\} - \min_j \left\{ EAC(K_i^{\text{sha}}, Q_i) \cdot EAC(K_i^{\text{mer}}), AM_{ij}^{\text{scr}} \right\},$$ (15)

$$\text{Noise} = \max_j \left\{ EAC(K_i^{\text{sha}}, Q_i) \cdot \left[ EAC(K_i^{\text{dec}}), AM_{ij}^{\text{sha}} \right] - EAC(K_i^{\text{mer}}), AM_{ij}^{\text{scr}} \right\} \right\} - \min_j \left\{ EAC(K_i^{\text{sha}}, Q_i) \cdot \left[ EAC(K_i^{\text{dec}}), AM_{ij}^{\text{sha}} \right] - EAC(K_i^{\text{mer}}), AM_{ij}^{\text{scr}} \right\}. \right\}$$ (16)

Signal measures information reserved in the shared AM, while noise measures the remaining part. Noise results from interference from other tasks. In Fig. 12, the case with $\beta = 1$ can be viewed as a particular case of DREAMS, corresponding to IP-HDC [19]. Meanwhile, the other cases drop $D \times (1 - \beta)$ inconsequential dimensions before AM sharing. The results show that IP-HDC reserves the most information; however, interference is also significant, leading to noisy similarity checking. On the other hand, two degrees of freedom make DREAMS strike the right balance between information and interference. Therefore, DREAMS is more flexible than IP-HDC in terms of utilization of the model capacity, leading to a 6.91% of accuracy improvement (the case with $\beta = 0.75$).
Fig. 13 compares the performance of DREAMS and IP-HDC at different $D_{sha}$. They have negligible difference for $D_{sha} < 4K$. However, for IP-HDC, when the size of the HD space $D_{sha}$ grows, interference grows faster than information. The severe interference hinders IP-HDC from efficiently exploiting model capacity. Therefore, a larger HD space does not improve the performance of IP-HDC. Instead, it leads to worse performance than a smaller HD space, as shown in Fig. 13a.

In contrast, DREAMS is more robust when $D_{sha} > 4K$. For DREAMS, information and interference both grow slower, and the two-stage mechanism keeps information larger than interference steadily. Therefore, DREAMS utilizes the model capacity more efficiently, outperforming IP-HDC by 11.74% in accuracy ($D_{sha} = 8.5K$).

### 5.3 Evaluation of DRIMER

We compare DRIMER with the state-of-the-art compression methods, including CompHD [25] and SparseHD [20], on the Permuted UCI-HAR dataset. The number of tasks and classes are set as $T = 5$ and $C = 6$, respectively. While $D$ is fixed at 10K, these methods aim to minimize the retained number of dimensions $D_{retain}$ in each compressed AM while preserving accuracy.

The results are shown in Fig. 14. First, the performance of CompHD degrades drastically with lower $D_{retain}$, since its effectiveness for noise mitigation is based on high-dimensional spaces. Next, DRIMER and SparseHD have a negligible difference for $D_{retain} > 3K$, but DRIMER shows better robustness in low $D_{retain}$. For $D_{retain} = 1K$, DRIMER provides a 3.3% of accuracy improvement. Note that there is a critical point ($D_{retain} = 1K$) for DRIMER, where lower $D_{retain}$ leads to a severe accuracy drop since the retained information is not enough for accurate classification.

### 5.4 Evaluation of MulTa-HDC Framework

In this section, we demonstrate the scalability of MulTa-HDC, we conduct experiments over the “Permuted” task protocol. For comparison purposes, apart from the shared-AM and independent-AM approaches, we introduce the ideal benchmark and baseline method described in [19]. For the ideal benchmark, the complete AM table is stored, leading to the highest accuracy at the highest storage space costs. On the contrary, the baseline method directly averages class HVs of the same class in the AM table:

$$AM_j = \sum_{i=1}^{T} AM_{i,j}$$

where $j = 1, \ldots, C$. The baseline method causes severe interference between tasks, leading to the lowest accuracy.

To evaluate the effectiveness of MulTa-HDC in accommodating a growing number of tasks $T$, we set $T = 1, 2, \ldots, 20$, corresponding to the application scenarios of edge devices with memory resources ranging from abundant to highly-constrained. For each method, the storage requirement for AM is fixed at $CD$, where $C$ is the number of classes and $D$ is the dimensions in HV, by parameter tuning. This fixed storage requirement is exactly the memory overhead of the baseline method. The MTL system aims at accommodating more tasks while preserving classification accuracy. We present experimental results in Fig. 15.
5.4.1 Application Scenario 1: Resource-Abundant Devices

In the scenario of edge devices with abundant memory resources, we consider an MTL system with a small number of tasks \( T < 12 \). As shown in Fig. 15, DRIMER consistently outperforms other two independent-AM approaches. For the Permuted UCIHAR in \( T = 11 \), DRIMER provides 3.83% and 6.67% accuracy improvements compared with SparseHD and CompHD, respectively. This is because DRIMER can retain more significant dimensions for classification by considering the label information in the training set. The performance of CompHD drastically degrades when \( T \) grows. Since larger \( T \) results in fewer dimensions retained by each independent AM, the orthogonality in HD space does not guarantee the noise reduction. In this scenario, MulTa-HDC sets \( \alpha = 1 \) and assign all of the dimensions to independent AMs, corresponding to DRIMER.

5.4.2 Application Scenario 2: Resource-Limited Devices

The performances of DRIMER and SparseHD are stable in the previous scenario. However, in real-world scenarios, each task would not be granted many memory resources. MulTa-HDC can leverage DREAMS to accommodate more discriminative information to deploy models more reliably and cost-effectively. As shown in Figs. 15a and 15b, the gap between the accuracy of DREAMS and DRIMER is closed as \( T \) increases. For example, in the Permuted MNIST, gaps between the accuracy provided by DRIMER and DREAMS are 2.89% \( (T = 5) \) and 1.36% \( (T = 20) \), respectively. DREAMS even outperforms DRIMER on the Permuted UCIHAR dataset when \( T > 16 \), implying that the degree of positive effects brought by more information reserved for each task is higher than that of negative impacts brought by interference. By leveraging both AM approaches, MulTa-HDC can adaptively adjust the three ratios \( \alpha, \beta, \) and \( \gamma \) to achieve the best performance in both scenarios. When memory resources are abundant, corresponding to the application scenario 1, higher \( \alpha \) prevents interference between tasks, leading to stable classification accuracy. Lower \( \alpha \) in the system implies more dimensions are reserved for each task in the shared AM. Therefore, MulTa-HDC can leverage the combined strengths of independent and shared AMs. Based on the experimental results, MulTa-HDC mitigates the performance degradation when the resources on edge devices are highly constrained. In the Permuted UCIHAR tasks when \( T = 20 \), MulTa-HDC uses the configuration of \( (\alpha, \beta, \gamma) = (0.4, 0.95, 0.05) \) to achieve 6.95% and 8.23% of accuracy improvements, compared with SparseHD and CompHD, respectively.

5.4.3 MTL-HDC System With Datasets From Heterogeneous Domains

To further demonstrate the generality of MulTa-HDC, we conduct experiments over “Split MNIST” task protocol, which is adopted in many MTL-related works [2], [26], [27]. It requires learning to classify MNIST digits sequentially. In Split MNIST, we split 10 digits into 2 disjoint sets, each of which corresponds to a subtask that contains 5 labels. Split ISOLET and Split UCIHAR share the same concept as Split MNIST.

In Split ISOLET, 26 labels are split into 8 disjoint sets, each of which corresponds to a subtask that contains 3 labels. Split ISOLET randomly drops two labels since the number of classes is not divisible by 3. Similarly, in Split UCIHAR, there are 3 subtasks, and each subtask contains 2 labels. The MTL-HDC system starts with learning 8 subtasks in Split ISOLET, corresponding to the section with blue background color in Fig. 16. After that, the system sequentially accommodates the subtasks in Split MNIST (green section in Fig. 16) and Split UCIHAR (brown section in Fig. 16). The HDC setup and evaluated datasets are summarized in Table 5.

Fig. 16 presents the experimental results. The x-axis denotes the number of subtasks in the MTL-HDC system. Note that the dotted black line indicates the ideal benchmark, which is the accuracy upper bound. Fig. 16 shows that MulTa-HDC performs slightly worse than the ideal benchmark, but better than other methods. While SparseHD and CompHD demonstrate dramatic improvements in memory efficiency of the MTL-HDC system, they still suffer from significant drop in accuracy. Instead, MulTa-HDC consistently outperforms prior works and shows scalability when the number of subtasks grows. Note that even in the transitions of training procedure of subtasks from heterogeneous domains (speech, image, and human activity), MulTa-HDC still preserves classification accuracy. The results imply that MulTa-HDC can not only accommodate datasets from multiple domains, but also handle multi-task transitions of training procedure of subtasks.
scenarios where the subtasks do not have the same number of labels. Based on these experiments, MulTa-HDC can efficiently handle both protocols defined in [26], including the “Split” and the “Permuted” task protocols.

6 ANALYSIS OF COMPLEXITY

6.1 Memory Complexity

This section provides quantitative evidence that MulTa-HDC achieves an efficient trade-off between memory overhead and performance degradation. In the training phase of MulTa-HDC, it finds task-specific mask HVs to encode task information and store the AMs compactly. In the inference phase, mask HVs act as auxiliary HVs to decode task-specific information from AMs with EAC operations, as shown in Algorithm 3 and 6, before Hamming distance computation. Apart from AMs, we should consider the extra storage space for the mask HVs.

To analyze the memory complexity of MulTa-HDC, we conduct experiments on Permuted UCIHAR with 12 tasks and calculate the storage requirement. The experimental setups are summarized in Table 6. Except for the ideal benchmark, we fine-tuned the parameters of all methods to fix the storage requirement for AMs at 60Kb (10K bits for each of the 6 classes in UCIHAR, not including the storage space for auxiliary HVs). Specifically, sparsity ratio \(s\) of SparseHD equals \((1 - \frac{1}{T})\), since SparseHD reduces the storage overhead of a single task from \(C\) to \(C(1 - s)\). Similarly, compression factor \(r\) of CompHD is exactly equal to \(T\), since CompHD reduces the storage overhead of a single task from \(CD\) to \(C\). As for parameters settings of MulTa-HDC, we follow the optimization problem described in Equation (6) of Section 3.1.

Table 7 shows the memory breakdown analysis, including the memory complexity of AM and auxiliary HVs for additional processing before computing the Hamming distance. For MulTa-HDC and SparseHD, these auxiliary HVs are mask HVs. As for CompHD, these auxiliary HVs are mutually orthogonal modulating HVs. In Table 7, \(S\) and \(r\) denote the sparsity ratio of SparseHD and the compression factor of CompHD, respectively.

Table 7 shows that while the ideal benchmark achieves the best performance at the highest memory overhead cost, the baseline method shows the opposite behavior. Neither of them provides satisfactory performance in real-world applications. Conversely, SparseHD and CompHD demonstrate dramatic improvements in memory efficiency. However, there is still an over 9% accuracy drop compared to the ideal benchmark. Meanwhile, MulTa-HDC can trade less than 5% accuracy drop for saving 77% memory overhead, achieving an efficient trade-off. Note that the mask HVs of MulTa-HDC are highly sparse, and thus many off-the-shelf compression techniques are applicable. In this case, the ratio of 1’s in mask HVs is around 5%, and we estimate the required storage of mask HVs can be reduced from 302 Kb to 86.4 Kb [30].

6.2 Discussion of Computational Complexity

When analyzing the computational complexity of MulTa-HDC, the extra processing for decoding should be considered. Table 8 shows the approximated estimation of the computational complexity in terms of bit-level operation counts for one classification. Note that the computational analysis of EAC operations in MulTa-HDC is not straightforward due to its architecture-dependent implementation. Therefore, we use EAC(\(x\)) to denote the bit-level operation count for EAC with a \(x\)-dimension input HV. The execution time is also reported based on an AMD Ryzen Threadripper 2990WX 32-core processor.

All methods in Table 8 have the same computational requirement for HDC encoding, as shown in Equation (1). Meanwhile, the encoding process accounts for more than 95% energy consumption in a binary HDC model during inference [31]. To present a clear comparison, we only
consider the computation required after the query HV has been encoded. Table 8 shows that the execution time of MultiTa-HDC is the highest. However, the crucial idea here is that the encoding module is the main computational bottleneck. Therefore, the computational overhead caused by the decoding in MultiTa-HDC is negligible.

7 Conclusion

This paper proposes a hybrid MultiTa-HDC framework for MTL-HDC based on shared-AM and independent-AM approaches. Proposed DREAMS shows higher utilization of model capacity for the shared-AM approach while outperforming the prior work by 11.74% of accuracy. For the independent-AM approach, proposed DRIMER compresses AMs in a supervised manner and provides a 3.3% of accuracy improvement compared with SparseHD. Moreover, MultiTa-HDC helps develop a more flexible framework for MTL-HDC by adjusting the proportion of the independent AMs and the shared AM depending on the memory capacity on device. Simulation results show that MultiTa-HDC can realize the best trade-off between memory overhead and performance degradation, making the MTL-HDC system suitable for the future of IoT.

Acknowledgment

This work was supported in part by the Ministry of Science and Technology of Taiwan under Grant MOST 109-2221-E-002-175. Cheng-Yang Chang and Yu-Chuan Chuang contributed equally to this work.

References


TABLE 8

<table>
<thead>
<tr>
<th>Framework</th>
<th>Bit-level Operation Count</th>
<th>Execution Time [μs]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ideal Benchmark</td>
<td>C(D + popcount(D))</td>
<td>79.4</td>
</tr>
<tr>
<td>MultiTa-HDC</td>
<td>2EAC(D) + C(abD + popcount(abD))</td>
<td>615.7</td>
</tr>
<tr>
<td>SparseHD</td>
<td>EAC(D) + C(D + popcount(D))</td>
<td>307.3</td>
</tr>
<tr>
<td>CompH</td>
<td>D + D/3 + popcount(r) + C(D + popcount(D))</td>
<td>245.6</td>
</tr>
<tr>
<td>Baseline</td>
<td>C(D + popcount(D))</td>
<td>79.4</td>
</tr>
</tbody>
</table>

* EAC(x) denotes the operation count for EAC with a x-dimension input.
† popcount(x) denotes the bit-level operation count required for counting the number of 1’s in a x-bit sequence.
Yu-Chuan Chuang (Student Member, IEEE) received the BS degree in electronic engineering from National Taiwan University, Taipei, Taiwan, in 2018, where he is currently working toward the PhD degree at the Graduate Institute of Electronics Engineering. His research interests include hyperdimensional computing, low-power design, and VLSI implementation.

En-Jui Chang (Member, IEEE) received the PhD degree from the Graduate Institute of Electronics Engineering, National Taiwan University, Taiwan, in 2014. From 2014 to 2017, he was a senior engineer with MediaTek Inc., Taiwan and worked on the development of LTE baseband modem algorithms and architectures. From 2017 to 2018, he was a postdoctoral fellow with Intel-NTU IoX Center and worked on machine learning and emotion classification. From 2018 to 2019, he was a postdoctoral fellow with ETH Zurich, Switzerland and worked on hyperdimensional computing development. He is currently a hardware architect of AI system with MediaTek Inc., Taiwan.

An-Yeu (Andy) Wu (Fellow, IEEE) received the BS degree in electrical engineering from NTU in 1987 and the MS and PhD degrees in electrical engineering from the University of Maryland, College Park, in 1992 and 1995, respectively. In August 2000, he was with the faculty of the Department of Electrical Engineering and the Graduate Institute of Electronics Engineering, NTU, where he is currently a distinguished professor. His research interests include low-power or high-performance VLSI architectures and IC designs for DSP or communication or AI applications, and adaptive or biomedical signal processing. In 2015, he was promoted to IEEE fellow for his contributions to DSP algorithms and VLSI designs for communication IC/SoC. From 2016 to 2019, he was the director of the Graduate Institute of Electronics Engineering, National Taiwan University. He is currently the editor-in-chief of the IEEE Journal on Emerging and Selected Topics in Circuits and Systems.

Cheng-Yang Chang (Student Member, IEEE) received the BS degree in electronic engineering from National Taiwan University, Taipei, Taiwan, in 2019, where he is currently working toward the PhD degree at the Graduate Institute of Electronics Engineering. His research interests include hyperdimensional computing, multi-task learning, and collaborative learning.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.