HyperDyn: Dynamic Dimensional Masking for Efficient Hyper-Dimensional Computing

Fangxin Liu[1,2,†], Haomin Li[1,2,†], Zongwu Wang[1,2], Dongxu Lyu[1], Li Jiang[1,2]
1. Shanghai Jiao Tong University, 2. Shanghai Qi Zhi Institute
haominli@sjtu.edu.cn, liufangxin@sjtu.edu.cn, ljiang_cs@sjtu.edu.cn

***Abstract*—Hyper-dimensional computing (HDC) is a bio-inspired computing paradigm that mimics cognitive tasks by encoding data into high-dimensional vectors and employing non-complex learning techniques. However, existing HDC solutions face a major challenge hindering their deployment on low-power embedded devices: the costly associative search module, especially in high-precision computations. This module involves calculating the distance between class vectors and query vectors, as well as sorting distances. In this paper, we present HyperDyn, an efficient dynamic inference framework designed for accurate and efficient hyper-dimensional computing. Our framework first offline analyzes the importance of different dimensions in the associative memory based on the contributions of the dimensions to the classification accuracy. In addition, we introduce a dynamic dimensional importance scaling mechanism for more flexible and accurate dimension contribution judgments. Finally, HyperDyn achieves efficient dynamic associative search through a dimension masking mechanism that adapts to the characteristics of the input sample. We evaluate HyperDyn on datasets from three different fields and the results show that HyperDyn can achieve $7.65\times$ speedup and 58% energy savings, with less than 0.2% loss in accuracy.**

***Index Terms*—Hyper-Dimensional Computing, Brain-inspired Computing, Dynamic Inference**

I. Introduction

As the Internet of Things (IoT) continues to evolve, the deployment of machine learning algorithms for cognitive tasks such as speech recognition and image classification has become increasingly common. However, the substantial computational complexity and memory demands of existing deep learning algorithms present obstacles for their widespread use in real-world embedded applications, particularly those constrained by limited device resources and power budgets [1]. Consequently, there is a growing need for alternative learning methods that can operate effectively on less powerful IoT devices while still delivering satisfactory classification accuracy [2]–[7].

Hyper-dimensional computing (HDC) [8] is an emerging paradigm in brain-like computing, utilizing vectors in hyper-dimensional space, typically exceeding 1,000 dimensions, to emulate neuronal activation patterns in the human brain. HDC excels in providing powerful representations and efficient learning capabilities, making it particularly suitable for edge computing applications [9]–[18]. In HDC, hardware-friendly operations are employed to encode additional information based on hyper-dimensional vectors. The inference process involves an associative search with class vectors created during training. This simplicity in operations and associative search results in HDC models with exceptionally low computational overhead and high parallelism [19], [20], making HDC well-suited for scenarios demanding minimal power consumption and low latency.

TABLE I
Energy Consumption Evaluation on HDC Inference.

Tasks	Energy Consumption (μJ)		Accuracy (%)
	Encode	Associative Search	
Speech Recognition [30]	2.115	6.345	90.51
Activity Detection [31]	1.905	2.540	92.63
Disease Classification [32]	0.093	0.465	78.87

Despite their immense potential, there is a need for low energy consumption and near-instantaneous inference in edge scenarios. Consequently, substantial efforts have been devoted to accelerating HDC models to achieve minimal latency. From the algorithm aspects, common strategies involve quantization [21], [22], which reduces the bit width of vector elements to simplify computational units, and dimensionality sparsification [23]–[27], aiming to decrease vector dimensionality and, consequently, computational unit executions for accelerated inference. From the hardware aspects, they incorporate efficient hardware designs with HDC, including in-memory computing [20], [28], and FPGA implementations [29]. However, these techniques are not optimal for accelerating HDCs due to two primary reasons. First, existing HDC algorithms use a static encoder without the capability to dynamically evaluate the importance of dimensions. Second, HDCs still require a large dimensionality to capture all potential relations between input features. This motivates us to explore a key question: *Can we drop dimensions on the fly to speedup HDC execution without sacrificing accuracy?*

To identify HDC's inference bottleneck, we evaluate accuracy and energy consumption breakdown on three tasks[1] (see Table I). Our analysis shows that associative search overhead is nearly 3.05 times higher than encoding. Encoding involves lookup and vector addition, while associative search contains many multiplications, becomes a crucial target for dynamic acceleration.

† These authors contributed equally. This work was sponsored by National Natural Science Foundation of China (Grant No.62402311) and Natural Science Foundation of Shanghai (Grant No.24ZR1433700). Li Jiang is the corresponding author.

[1]Evaluation on FPGA with one sample inference, HDC model configured with dim=1,000 and quantized to 8 bits.

978-3-9826741-0-0/DATE25/© 2025 EDAA

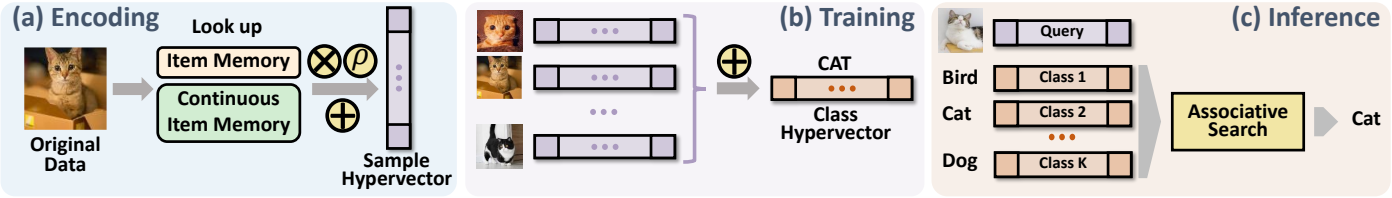

Fig. 1. Hyper-Dimensional Computing Basics. HV is short for hypervector. (a) Encoding: Raw data is represented in a high-dimensional space using item memory (IM) and continuous item memory (CIM). (b) Training: Class hypervectors are created by bundling all sample vectors belonging to the same class. (c) Inference: Associative search is performed to match the query hypervector with the stored class hypervectors, resulting in a prediction.

In this paper, we present HyperDyn, an efficient dynamic inference framework for HDC. We first introduce a novel fuzzy-based measure for dimensional importance in associative memory, leveraging fuzzy logic and a detailed analysis of the computational flow in associative search. To enhance precision, we propose a dynamic dimensional scaling strategy based on input query hyper-dimensional vectors, informed by a comprehensive study of similarity computation during inference. Additionally, a masking mechanism is incorporated to streamline efficient associative search. The proposed HyperDyn framework is validated through a hardware implementation, highlighting its practical utility. The main contributions of this paper are summarized as follows:

- We analyze the dimensional importance in associative search, and propose a fuzzy-based measure for the vector dimension. This measure serves as an effective tool for assessing the importance of dimensions, guiding the computation reduction.
- Building on the dimension importance analysis, we introduce a dynamic strategy that scales dimensional importance based on input query values. This approach minimizes accuracy loss during the reduction of dimensions in HDCs.
- We present a hardware-efficient dynamic HDC framework alongside a corresponding hardware implementation. This framework enables the adaptive reduction of vector dimension through dynamic masking based on the input sample.

II. Backgrounds

A. Hyper-Dimensional Computing

HDC is a brain-inspired computing paradigm that represents and learns data using vectors in hyper-dimensional space, referred to as hypervectors. As shown in Figure 1, HDC consists of three core modules: Item Memory (IM), Continuous Item Memory (CIM) [33], and Associative Memory (AM), unfolding across three learning stages: encoding, training, and inference.

Encoding: In this stage, HDC maps each element of atom data from a sample to a base hypervector using IM and CIM. IM stores binary base hypervectors for discrete values, while CIM stores those for continuous values. HDC then encodes the mapped basic hypervectors into a sample hypervector through operations: (i) Bundling $\oplus$: Element-wise addition on multiple hypervectors results in a single hypervector.(ii) Binding $\otimes$: Element-wise XOR is performed on two hypervectors. (iii) Permutation ρ: A rotational shift over a single hypervector is applied.

To map a feature to a space, HDC looks up one hypervector in IM and one in CIM. The base hypervector from IM may be permuted to encode location information. These two hypervectors are then bound to complete the semantic combination, and the hypervectors computed from each feature in a feature vector are bundled together to form the sample hypervector.

Training: In the training phase, HDC employs the bundling operation to aggregate hyper-dimensional representations of samples within the same class, generating class hypervectors stored in AM. Some frameworks enhance accuracy by retraining the model through multiple iterations on the training data. Similar to the inference stage, the model attempts to predict the label of the data. For mispredicted samples, their hypervectors are subtracted from the mispredicted class and added to the correct class hypervectors based on the ground truth.

Inference: During inference, HDC searches the AM for class hypervectors most similar to the query hypervector encoded from the query data. The identified class is then predicted. HDC employs cosine distance as the similarity measure and hamming distance for binary models.

B. Dynamic Inference

In this section, we briefly introduce previous dynamic DNNs and emphasize the novelty of our method compared with them. Dynamic inference can be broadly classified into two main categories: dynamic architecture and dynamic parameters [34], [35]. Dynamic architectures involve techniques like early exiting, skipping specific layers, and dynamic routing. On the other hand, dynamic parameters approaches include operations like dynamic sparsity. Generally, dynamic inference methods provide powerful capability in reducing computational redundancy, by automatically adjusting their architectures for different inputs. However, challenges exist: (1) dynamic architectures, while directly reducing computation, may introduce extra burden on hardware due to dynamic determinations, and (2) dynamic parameter approaches face challenges related to index overhead and compute/access irregularity, making practical benefits often lag behind theoretical advantages.

For HDC, a promising computing platform, several efforts based on dynamic inference aim to enhance performance. AdaptBit-HD [36] uses dynamic quantization, starting from the highest bit of vectors in associative memory and participating in similarity computation bit by bit. YC Chuang, et al. [26]

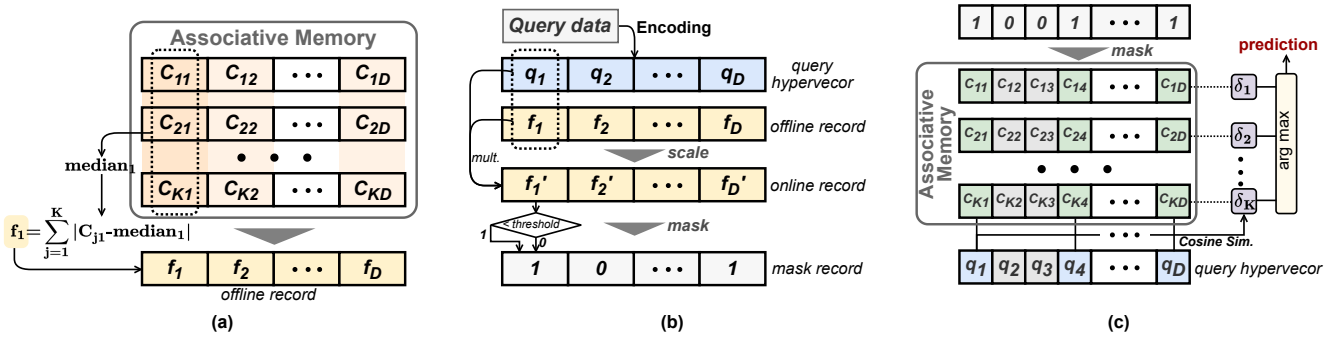

Fig. 2. HyperDyn framework. (a) Associative Memory Dimensional Importance Generation. (b) Dynamic Dimensional Scaling & Masking. (c) Similarity Calculation and Prediction.

uses binary HDC for initial inference and switches to the INT-based model if the confidence is insufficient. The size of the dimension range in the INT model's associative search is determined by the confidence interval. On the other hand, MHD [27] pre-trains multiple models of different dimensions and employs an HDC model for routing model selection based on each model's proficiency in classifying specific samples. However, the latency cost associated with the architectural design of multi-stage models erases the advantage of dynamic inference.

Our method differs from previous dynamic HDCs in the following three aspects: 1) We propose fine-grained dynamic dimension selection to improve hardware efficiency. 2) We introduce a dynamic scaling to automatically adjust importance thresholding for different inputs. 3) We design a hardware-friendly masking mechanism to efficiently filter out less crucial dimensions.

C. Motivation

In practice, many dynamic inference methods are implemented as inefficient path indexing or routing, creating a significant gap between theoretical analysis and practical acceleration. This challenge also exists in HDC inference. The previous bit-level dynamic scheme [36] is constrained to bit-serial computing, resulting in significantly higher inference latency for high-precision computation. Besides the bit-width of hypervectors, alternative methods target dynamic routing among different paths in multi-model scenarios, aiming for coarse-grained computational savings. Despite these efforts, existing dynamic HDC methods still struggle with the performance-complexity trade-off. Therefore, we propose an adaptive dynamic HDC framework: dynamically masking dimensions based on their importance to reduce real-time computation. We posit that such a work presents an opportunity to unlock the full potential of dynamic inference in HDC, offering a promising avenue for enhanced performance without sacrificing model accuracy.

III. HyperDyn Framework

A. overview

This section presents HyperDyn, an efficient HDC execution framework tailored for hardware-efficient dynamic inference. The HyperDyn framework can be depicted in Figure 2. We first derive the dimensional importance through offline dimensional analysis of associative memory. Then, an adaptive dimension scaling scheme is performed to get input-aware dimensional importance, thereby enhancing overall accuracy. Finally, we employ dynamic HDC and implement dimension masking based on the adjusted importance, effectively accelerating associative searches within HDCs.

B. Associative Memory Dimensional Analysis

To facilitate dynamic dimension selection, we develop a dimensional importance measuring scheme for HDC. We begin by analyzing the associative search process within associative memory. During this search, HDC conducts cosine similarity checks between the input query hypervector ($\vec{S}$) and the hypervectors ($\vec{C_i}$) of K classes in AM. The class exhibiting the maximum similarity is chosen as the prediction result. The calculation can be formulated as Equation 1:

$$prediction = \underset{i=1,\dots,K}{\arg\max}\ Cos(\vec{S},\vec{C_i}) = \frac{\vec{S}\cdot\vec{C_i}}{|\vec{S}||\vec{C_i}|} \tag{1}$$

Here, $\vec{S}$ represents the sample hypervector, and $\vec{C_i}$ denotes the hypervector of class i. Recognizing that our focus is on the relative magnitude of similarities, we can disregard the term $\frac{1}{|\vec{S}|}$. Furthermore, if all class hypervectors are normalized, the term $\frac{1}{|\vec{C_i}|}$ can also be omitted. Consequently, the cosine similarity simplifies to the dot product. Our goal is to express the calculation, preceding the identification of the maximum value, as a vector matrix multiplication:

$$\begin{aligned}\arg\max \begin{bmatrix} Cos(\vec{S},\vec{C_1}) \\ Cos(\vec{S},\vec{C_2}) \\ \dots \\ Cos(\vec{S},\vec{C_K}) \end{bmatrix} &= \arg\max \begin{bmatrix} \vec{C_1} \\ \vec{C_2} \\ \dots \\ \vec{C_K} \end{bmatrix} \cdot \vec{S} \\ = \arg\max \begin{bmatrix} C_{11} & C_{12} & \dots & C_{1D} \\ C_{21} & C_{22} & \dots & C_{2D} \\ & & \dots & \\ C_{K1} & C_{K2} & \dots & C_{KD} \end{bmatrix} &\begin{bmatrix} S_1 \\ S_2 \\ \dots \\ S_D \end{bmatrix}\end{aligned} \tag{2}$$

where $\vec{C_i}$ is the row vector, $\vec{S}$ is the column vector, $\{C_{i1},\dots,C_{i_D}\}$ represents the components of $\vec{C_i}$, and $\{S_1,\dots,S_D\}$ denotes the components of $\vec{S}$.

For a given $\vec{S}$, the impact of each column on the final prediction relies on the disparities among its elements. Consider an extreme scenario: when all elements within a column are

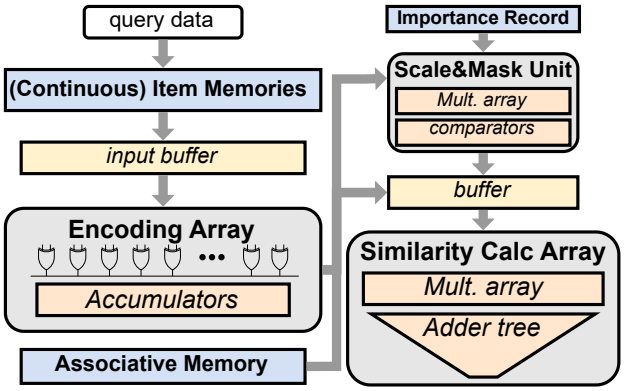

Fig. 3. Hardware Implementation in HyperDyn.

equal, that column has no impact on the final result and can be deemed a redundant dimension. Consequently, the higher the discriminative contribution of columns (i.e., dimensions) to classification, the more crucial they become. Previous works, such as HyperAttack [37], employed a distribution-based approach and a fuzzy target-based Hamming distance as dimensional importance measures for binary HDCs. Similarly, ReHD [25] used absolute value and variance as measures.

In this paper, we adapt the fuzzy target-based measure from binary HDCs and extend it to non-binary associative memory. We derive the importance of a column by considering the fuzzing dimension case, measuring the importance of a corresponding dimension as the minimum edit distance required to fuzz all elements within a column to the same value—referred to as fuzzing distance. Specifically, for a column with elements $C_{1i}, C_{2i}, \ldots, C_{Ki}$ of the ith dimension, assuming the median of these numbers is $median_i$, the importance can be measured as follows:

$$f_i = \sum_{j=1}^{K} |C_{ji} - median_i| \tag{3}$$

Here, f_i denotes the fuzzing distance of the ith dimension. Given that models deployed on devices are typically quantized, importance measurement is performed on the quantized associative memory. As depicted in Figure 2 (a), we perform offline computation of the fuzzy target-based importance for each dimension in the associative memory and record the results.

C. Dynamic Dimensional Scaling & Masking

With the offline-recorded dimensional importance, we seamlessly deploy it to the device alongside the model, preparing for dynamic inference. During inference, HDC starts by encoding the query data into a query hypervector. For associative search, the similarities between the query hypervector and the class hypervectors in the associative memory are calculated, as outlined in Eq 1. Then, we observe that the offline-computed contribution of each dimension for classification is dynamically scaled by the value of the corresponding dimension in the query hypervector. Specifically, the similarity calculation of the query hypervector $\vec{S}$ with class i and class j unfolds as follows:

$$\begin{aligned} similarity_i &= C_{i1}S_1 + C_{i2}S_2 + \cdots + C_{iD}S_D \\ similarity_j &= C_{j1}S_1 + C_{j2}S_2 + \cdots + C_{jD}S_D \end{aligned} \tag{4}$$

If the xth dimension is crucial for distinguishing between these two classes, the difference between C_{ix} and C_{jx} is generally significant. However, if $\vec{S}$ does not belong to either class i or class j, S_x may be small, diminishing the contribution of the xth dimension to the query's classification. In other words, S_x diminishes the importance of the xth dimension, and there are cases where the dimensional importance is scaled up.

Hence, as illustrated in Figure 2, after encoding the query, we scale the importance record f to f' using the query hypervector $\vec{S}$:

$$f' = \{f'_i \mid \ \forall\ i \in [1, D],\ f'_i = f_i \cdot S_i\} \tag{5}$$

Based on the scaled importance records, we generate masks for dimensions based on a predefined threshold t. Specifically, for a dimension with importance below the threshold t, we set its mask to 0, and set it to 1 otherwise:

$$Mask_i = \begin{cases} 0 & if\ f'_i < t \\ 1 & otherwise \end{cases} \tag{6}$$

D. Similarity Calculation and Prediction

After completing mask generation, we proceed to the similarity calculation between the query hypervector and all class hypervectors. Specifically, we compute the inner product between $\vec{S}$ and the C_is. We selectively skip dimensions with a mask of 0 and only consider dimensions with a mask of 1. This can be formulated as follows:

$$similarity_i = \sum_{j=1}^{D} Mask_j C_{ij} S_j \tag{7}$$

The class with maximum similarity to $\vec{S}$ serves as the output prediction. With the introduced online dimensional importance scaling, it is crucial to analyze the computational overhead. In the original associative search, there are $K \times D$ multiplications. The dimensional importance scaling introduces an additional D multiplications. However, as long as there are more than $\lceil \frac{D}{K} \rceil$ 0s in the mask of length D, the number of multiplications in the associative search will be less than $(K-1) \times D$. This ensures a positive gain in our design.

E. Hardware Implementation

To fully leverage the advantages of our framework, we have devised a hardware implementation for HyperDyn on the FPGA platform, as illustrated in Figure 3. The design encompasses item memory, associative memory, importance record buffer, encoding array, scale & mask unit, and similarity calculation unit.

- Item Memory and Associative Memory: Implemented using Block RAM.
- Buffers: Realized with Distributed RAM.
- Scaling Unit and Similarity Calculation Unit: Realized with DSPs.

In the encoding stage, base hypervectors corresponding to the query data are retrieved from the Item Memory array. Subsequently, they are fed into the encoding array to generate the query hypervector. Following this, the vector stored in the importance record buffer and the query hypervector are loaded

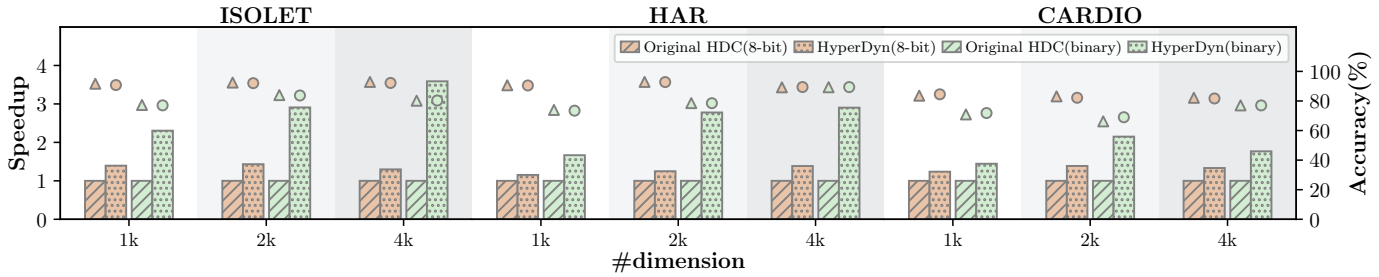

Fig. 4. Associative search speedup comparison of methods on 8-bit and binary HDC models.

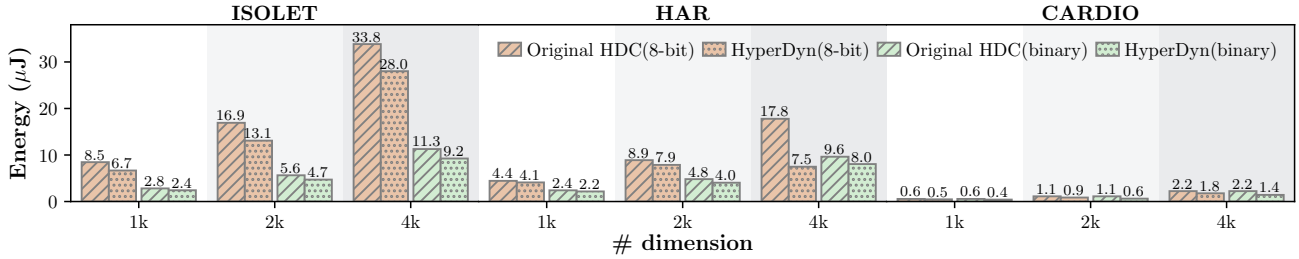

Fig. 5. Energy consumption comparison of methods on 8-bit and binary HDC models.

TABLE II
DATASET STATISTICS.

Dataset	#feature	#class	Train Size	Test Size	Description
ISOLET [30]	617	26	6,238	1,559	Speech recognition
HAR [31]	561	12	6,213	1,554	Activity recognition
CARDIO [32]	21	10	1,913	213	Disease classification

into the scaling unit to produce the scaled importance record. This record is then input into the mask unit to generate the binary mask.

During the associative search, relevant data, which includes the dimensions marked as 1 in the query hypervector and class hypervectors, is directed into the similarity computation unit. This unit incorporates element-wise multiplication arrays and adder trees to compute the similarities.

IV. EVALUATION

A. Experimental Setup

Platforms: We have implemented HyperDyn on both software and hardware platforms to evaluate its performance. In the software aspect, we developed the MIND Framework using Python and executed it on an Intel(R) Xeon(R) Silver 4208 CPU with 16GB memory. For hardware evaluation, we designed the architecture with heterogeneous computing cores on an embedded FPGA device, which is particularly suited for resource-constrained scenarios that demand high efficiency. The hardware implementation was carried out on the Kintex-7 FPGA using Verilog, with a working frequency set to 100MHz. To validate the timing and functionality of our models, we synthesized them using the Xilinx Vivado Design Suite [38] and obtained the energy result based on the Vivado power estimation report and the corresponding latency.

Benchmarks & Baselines: We evaluated our framework on three diverse datasets, the details of which are provided in Table II. Our main comparisons involved HyperDyn against the original HDC models, quantized to 8 bits and binary. All the models were evaluated with a dimension of $1,000$. We also explored different design parameters to analyze their impact on performance.

TABLE III
ACCURACY AND REDUCED COMPUTATION COMPARISON.

		Accuracy(%)				Reduced Computation(%)	
Methods		**Original HDC**		**HyperDyn**		**HyperDyn**	
Bit-widths		*8-bit*	*binary*	*8-bit*	*binary*	*8-bit*	*binary*
Datasets	**ISOLET**	90.51	74.28	90.44	73.96	15.40	66.62
	HAR	92.63	74.89	92.69	75.11	26.40	80.69
	CARDIO	78.87	60.56	78.87	60.56	32.40	56.70

B. Experimental Results

Accuracy & Speedup. Table III illustrates the impact of HyperDyn on accuracy and computational efficiency compared to the original HDC. Notably, for an INT8 model, HyperDyn achieves a significant 24.73% reduction in computation while ensuring a minimal average accuracy loss of 0.01%. On the CARDIO dataset, a remarkable 32.40% reduction in computation is attained without compromising original accuracy. HyperDyn also exhibits impressive efficiency on binary HDC models, achieving an average computation reduction of 68%. Particularly on the HAR dataset, computation is reduced by an impressive 80.69%, accompanied by a slight accuracy improvement of 0.22%. This improvement is attributed to HyperDyn's dynamic masking ability, notably effective in the binary HDC model, where it adeptly filters out interfering dimensions.

Figure 4 illustrates the speedup results for associative search across 1k, 2k, and 4k dimensional models. Notably, on the speech recognition task, an impressive speedup of $3.59\times$ is achieved for the 4k dimensional binary HDC. On average, across the three tasks, we achieve substantial speedups of $2.15\times$, $1.86\times$, and $1.55\times$, respectively. This acceleration is attributed to efficient dynamic masking, which strategically

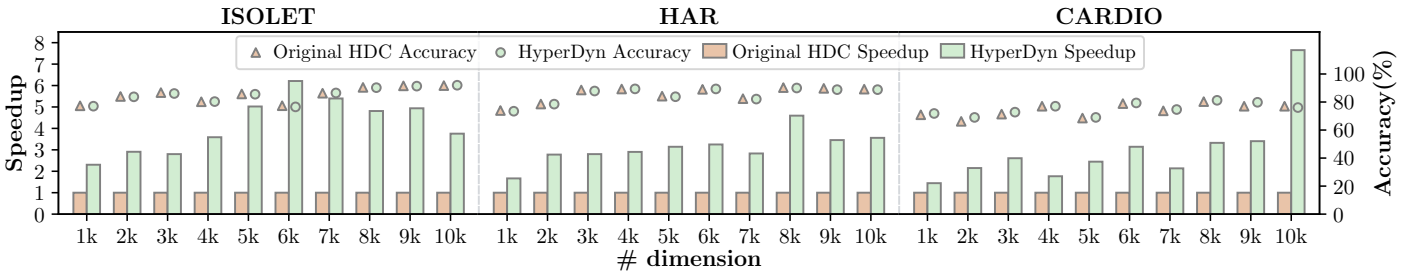

Fig. 6. Latency and accuracy comparison between original HDC and HyperDyn configured with different dimensionalities.

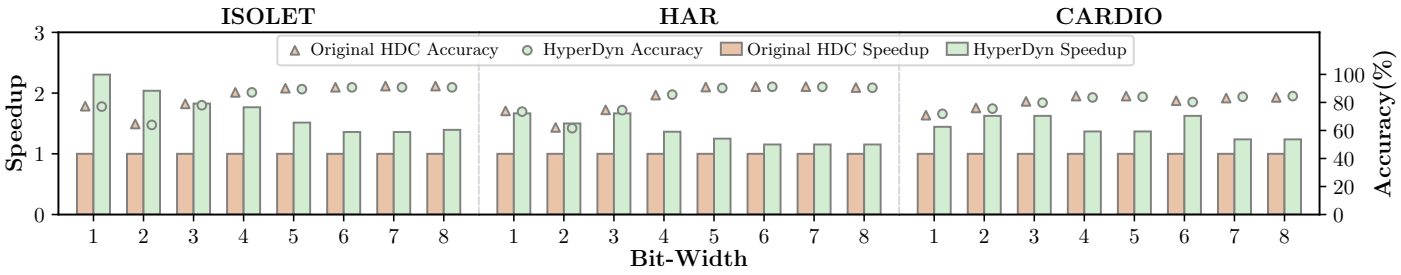

Fig. 7. Speedup and accuracy comparison between original HDC and HyperDyn over various bit-widths.

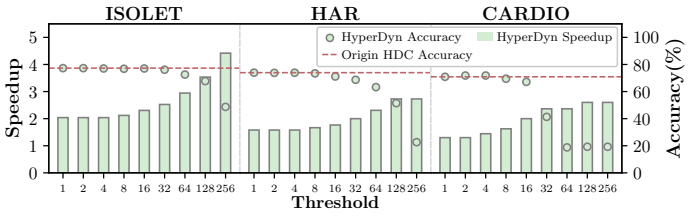

Fig. 8. Latency and accuracy of HyperDyn with different thresholds. The red dashed line indicates the accuracy of original HDC.

reducing computation and minimizing inference latency by filtering unimportant dimensions.

Energy Consumption. Figure 5 illustrates the energy consumption comparison between HyperDyn and the baseline. Notably, on the HAR dataset, we achieve a substantial 58% reduction in energy consumption for the INT8 HDC model with 4k dimensions. Even for the low-energy binary HDC model, HyperDyn achieves a notable $\approx$ 20% reduction in energy consumption.

Impact of Threshold. The threshold is a crucial factor in HyperDyn's masking mechanism, determining the number of calculated dimensions. In Figure 8, we observe its influence on HyperDyn's speedup and accuracy. The red dashed line depicts the accuracy of the original HDC. Remarkably, when the threshold is below 16, HyperDyn ensures minimal accuracy loss while achieving a substantial speedup ranging from 1.5 to 2.5 times. However, as the threshold increases, more dimensions are masked, leading to a noticeable decline in accuracy. This analysis provides valuable insights into the trade-off between accuracy and speedup, guided by the threshold parameter.

Impact of Dimension. Figure 6 studies the impact of dimensionality for binary HDC on three tasks. We sweep it from 1k to 10k. HyperDyn achivevs a peak speedup of 7.65× and solid average accelerations of 4.17×, 3.10×, and 3.01× across the respective tasks. Impressively, it incurs minimal accuracy losses—averaging only 0.2% and 0.32% on ISOLET and HAR datasets—while achieving an average accuracy improvement of about 1% on CARDIO. This highlights adaptability of HyperDyn to dimensionality changes, striking a fine balance between acceleration gains and accuracy.

Impact of Bit-width. In Figure 7, we explore the impact of the bit-width for the HDC model in terms of speedup and accuracy. Notably, HyperDyn achieves significant speedup at low bit-widths, including binary and ternary, reaching up to 2.3×. Importantly, HyperDyn consistently incurs a negligible accuracy loss ($\leq$ 1%) across all bit-widths. This highlights the effectiveness of HyperDyn in maintaining accuracy while delivering significant speedup benefits, particularly in scenarios requiring limited bit-widths.

V. Conclusion

In this paper, we introduce HyperDyn, an effective dynamic inference framework tailored for HDCs, aiming to reduce computation latency caused by associative searches in the hyper-dimensional space. We explore associative memory and present a method for measuring the importance of dimensions. Building upon the characteristics of similarity computation, we propose an online scaling and masking mechanism with its corresponding hardware implementation. This allows dynamic computation advantages without adding significant overhead. Our evaluations demonstrate that HyperDyn can deliver satisfactory performance gain with trivial accuracy loss so HyperDyn is promising to deploy the HDC models on edge devices. In the future, based on the insights from this work, we plan to explore the interactions between dimensions. This exploration aims to refine our importance analysis, enabling more precise and efficient steering of dynamic inference, paving the way for the development of a highly efficient accelerator with dynamic execution.

REFERENCES

[1] F. Liu, N. Yang, H. Li, Z. Wang, Z. Song, S. Pei, and L. Jiang, "Spark: Scalable and precision-aware acceleration of neural networks via efficient encoding," in *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2024, pp. 1029–1042.

[2] L. Deng, Y. Wu, Y. Hu, L. Liang, G. Li, X. Hu, Y. Ding, P. Li, and Y. Xie, "Comprehensive snn compression using admm optimization and activity regularization," *IEEE transactions on neural networks and learning systems*, 2021.

[3] H. Li, F. Liu, Z. Sun, Z. Wang, S. Huang, N. Yang, and L. Jiang, "Neuronquant: Accurate and efficient post-training quantization for spiking neural networks," in *2025 30th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2025.

[4] F. Liu, Z. Wang, W. Zhao, N. Yang, Y. Chen, S. Huang, H. Li, T. Yang, S. Pei, X. Liang *et al.*, "Exploiting temporal-unrolled parallelism for energy-efficient snn acceleration," *IEEE Transactions on Parallel & Distributed Systems*, no. 01, pp. 1–16, 2024.

[5] Z. Wang, F. Liu, N. Yang, S. Huang, H. Li, and L. Jiang, "Compass: Sram-based computing-in-memory snn accelerator with adaptive spike speculation," in *2024 57th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2024, pp. 1090–1106.

[6] F. Liu, H. Li, N. Yang, Z. Wang, T. Yang, and L. Jiang, "Teas: Exploiting spiking activity for temporal-wise adaptive spiking neural networks," in *2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2024, pp. 842–847.

[7] R. Yin, Y. Kim, D. Wu, and P. Panda, "Loas: Fully temporal-parallel dataflow for dual-sparse spiking neural networks," in *2024 57th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2024, pp. 1107–1121.

[8] P. Kanerva, "Hyperdimensional Computing: An Introduction to Computing in Distributed Representation with High-Dimensional Random Vectors," *Cognitive Computation*, vol. 1, 2009.

[9] D. Ma, T. S. Rosing, and X. Jiao, "Testing and enhancing adversarial robustness of hyperdimensional computing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2023.

[10] Z. Zou, H. Chen, P. Poduval, Y. Kim, M. Imani, E. Sadredini, R. Cammarota, and M. Imani, "Biohd: an efficient genome sequence search platform using hyperdimensional memorization," in *Proceedings of the 49th Annual International Symposium on Computer Architecture*, ser. ISCA '22, 2022, pp. 656–669.

[11] M. Imani, D. Kong, A. Rahimi, and T. Rosing, "Voicehd: Hyperdimensional computing for efficient speech recognition," in *2017 IEEE international conference on rebooting computing (ICRC)*. IEEE, 2017, pp. 1–8.

[12] I. Nunes, M. Heddes, T. Givargis, A. Nicolau, and A. Veidenbaum, "Graphhd: Efficient graph classification using hyperdimensional computing," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2022, pp. 1485–1490.

[13] S. Zhang, R. Wang, J. J. Zhang, A. Rahimi, and X. Jiao, "Assessing robustness of hyperdimensional computing against errors in associative memory," in *ASAP*, 2021, pp. 211–217.

[14] F. Liu, H. Li, N. Yang, Y. Chen, Z. Wang, T. Yang, and L. Jiang, "Paap-hd: Pim-assisted approximation for efficient hyper-dimensional computing," in *2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2024, pp. 46–51.

[15] H. Li, F. Liu, Y. Chen, and L. Jiang, "Hypernode: An efficient node classification framework using hyperdimensional computing," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. IEEE, 2023, pp. 1–9.

[16] H. Li=, F. Liu, Y. Chen, and L. Jiang, "Hyperfeel: An efficient federated learning framework using hyperdimensional computing," in *2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2024, pp. 716–721.

[17] F. Liu, H. Li, Y. Chen, T. Yang, and L. Jiang, "Hyperattack: An efficient attack framework for hyperdimensional computing," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2023, pp. 1–6.

[18] F. Liu, H. Li, X. Yang, and L. Jiang, "L3e-hd: A framework enabling efficient ensemble in high-dimensional space for language tasks," in *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2022, pp. 1844–1848.

[19] M. Imani, Z. Zou, S. Bosch, S. A. Rao, S. Salamat, V. Kumar, Y. Kim, and T. Rosing, "Revisiting hyperdimensional learning for fpga and low-power architectures," in *HPCA*, 2021.

[20] G. Karunaratne, M. Le Gallo, G. Cherubini, L. Benini, A. Rahimi, and A. Sebastian, "In-memory hyperdimensional computing," *Nature Electronics*, vol. 3, no. 6, pp. 327–337, 2020.

[21] M. Imani, S. Bosch, S. Datta, S. Ramakrishna, S. Salamat, J. M. Rabaey, and T. Rosing, "Quanthd: A quantization framework for hyperdimensional computing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 10, pp. 2268–2278, 2019.

[22] J. Morris, S. T. K. Set, G. Rosen, M. Imani, B. Aksanli, and T. Rosing, "Adaptbit-hd: Adaptive model bitwidth for hyperdimensional computing," in *ICCD*. IEEE, 2021, pp. 93–100.

[23] J. Morris, M. Imani, S. Bosch, A. Thomas, H. Shu, and T. Rosing, "Comphd: Efficient hyperdimensional computing using model compression," in *2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE, 2019, pp. 1–6.

[24] M. Imani, S. Salamat, B. Khaleghi, M. Samragh, F. Koushanfar, and T. Rosing, "Sparsehd: Algorithm-hardware co-optimization for efficient high-dimensional computing," in *2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 2019, pp. 190–198.

[25] J. Morris, R. Fernando, Y. Hao, M. Imani, B. Aksanli, and T. Rosing, "Locality-based encoder and model quantization for efficient hyperdimensional computing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 4, pp. 897–907, 2021.

[26] Y.-C. Chuang, C.-Y. Chang, and A.-Y. Wu, "Dynamic-hdc: A two-stage dynamic inference framework for brain-inspired hyperdimensional computing," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2023.

[27] M. Imani, C. Huang, D. Kong, and T. Rosing, "Hierarchical hyperdimensional computing for energy efficient classification," in *Proceedings of the 55th Annual Design Automation Conference*, 2018, pp. 1–6.

[28] H. Li, T. F. Wu, A. Rahimi, K.-S. Li, M. Rusch, C.-H. Lin, J.-L. Hsu, M. M. Sabry, S. B. Eryilmaz, J. Sohn *et al.*, "Hyperdimensional computing with 3d vrram in-memory kernels: Device-architecture co-design for energy-efficient, error-resilient language recognition," in *2016 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2016, pp. 16–1.

[29] S. Salamat, M. Imani, and T. Rosing, "Accelerating hyperdimensional computing on fpgas by exploiting computational reuse," *IEEE Transactions on Computers*, vol. 69, no. 8, pp. 1159–1171, 2020.

[30] R. Cole and M. Fanty, "ISOLET," UCI Machine Learning Repository, 1994, DOI: https://doi.org/10.24432/C51G69.

[31] J. Reyes-Ortiz, D. Anguita, L. Oneto, and X. Parra, "Smartphone-Based Recognition of Human Activities and Postural Transitions," UCI Machine Learning Repository, 2015, DOI: https://doi.org/10.24432/C54G7M.

[32] D. Campos and J. Bernardes, "Cardiotocography," UCI Machine Learning Repository, 2010, DOI: https://doi.org/10.24432/C51S4N.

[33] A. Rahimi, S. Benatti, P. Kanerva, L. Benini, and J. M. Rabaey, "Hyperdimensional biosignal processing: A case study for EMG-based hand gesture recognition," in *2016 IEEE International Conference on Rebooting Computing (ICRC)*. San Diego, CA, USA: IEEE, Oct. 2016, pp. 1–8.

[34] Y. Han, G. Huang, S. Song, L. Yang, H. Wang, and Y. Wang, "Dynamic neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 11, pp. 7436–7456, 2022.

[35] S. Laskaridis, A. Kouris, and N. D. Lane, "Adaptive inference through early-exit networks: Design, challenges and directions," in *Proceedings of the 5th International Workshop on Embedded and Mobile Deep Learning*, ser. EMDL'21. New York, NY, USA: Association for Computing Machinery, 2021, p. 1–6.

[36] J. Morris, S. T. K. Set, G. Rosen, M. Imani, B. Aksanli, and T. Rosing, "Adaptbit-hd: Adaptive model bitwidth for hyperdimensional computing," in *ICCD*, 2021, pp. 93–100.

[37] F. Liu, H. Li, Y. Chen, T. Yang, and L. Jiang, "Hyperattack: An efficient attack framework for hyperdimensional computing," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*, 2023, pp. 1–6.

[38] T. Feist, "Vivado design suite," *White Paper*, vol. 5, 2012.