2023 Design, Automation & Test in Europe Conference (DATE 2023) – Late Breaking Results

A Linear-Time, Optimization-Free, and Edge Device-Compatible Hypervector Encoding

Sercan Aygun*, M. Hassan Najafi*, and Mohsen Imani†
sercan.aygun@louisiana.edu, najafi@louisiana.edu, m.imani@uci.edu
*University of Louisiana at Lafayette, †University of California Irvine

***Abstract*—Hyperdimensional computing (HDC) offers a single-pass learning system by imitating the brain-like signal structure. HDC data structure is in random hypervector format for better orthogonality. Similarly, in bit-stream processing – aka stochastic computing– systems, low-discrepancy (LD) sequences are used for the efficient generation of uncorrelated bit-streams. However, LD-based hypervector generation has never been investigated before. This work studies the utilization of LD Sobol sequences as a promising alternative for encoding hypervectors. The new encoding technique achieves highly-accurate classification with a single-time training step without needing to iterate repeatedly over random rounds. The accuracy evaluations in an embedded environment exhibit a classification rate improvement of up to `9.79%` compared to the conventional random hypervector encoding.**

I. Introduction

Hyperdimensional computing (HDC) [1] is an emerging computing methodology that mimics brain-like learning with highly efficient and noise-tolerant computation. Since the primitive operations in HDC are in the order of logic gates and depend on hardware-light solutions such as `XOR`ing, shifting, dot product, and population count, HDC is suggested as a promising alternative to complex systems such as deep learning in resource-limited environments. In HDC systems, the primitive data unit is a vector consisting of $+1$ (logic-1 in memory) and -1 (logic-0 in memory) values. Up to $10,000$ bit long hypervectors are used when *encoding data* (Fig. 1 (a)). During *training*, HDC superimposes the encoding of signal values to create a composite representation of a phenomenon of interest known as a "class ($\mathbf{C}$) hypervector" (Fig. 1 (b)). During the inference, the nearest *similarity* (δ) search returns the class of the encoded query hypervector (Fig. 1 (c)).

In HDC systems with non-continuous data, hypervectors need to be orthogonal to each other. Randomly generated vectors are nearly-orthogonal. Generally, pseudo-random methods are utilized to provide this randomness. However, this directly affects the accuracy and may require hundreds to thousands of rounds to determine the best random vectors. Hypervector optimization may not always be possible and can affect the performance if the training phase is performed on edge devices. This work proposes a simple vector encoding approach with $O(D)$ complexity (D is the vector size) without using random functions or vector optimization methods. The proposed technique uses *Sobol sequences* [2] for high-quality hypervector encoding. The new encoding approach is lightweight and does not require an exhaustive search to find the best-performing vectors, yielding efficient training on edge devices.

The basic operations in HDC are multiplication ($\oplus$: logical `XOR`), addition (Σ: bitwise population count), and permutation (Π: shifting). These operations are invertible and have linear time complexity. HDC systems first encode data with a proper technique according to the classification or cognitive tasks. Spatial, temporal, and histogram-based encoding techniques are used in the literature [3]. Encoders are divided into (i) record-based and (ii) n-gram-based approaches [4], [5]. The record-based approaches assign level hypervectors ($\boldsymbol{L}$, e.g., pixel intensity values) and position hypervectors ($\boldsymbol{P}$, e.g., randomly generated vectors for pixel positions). Feature positions on data

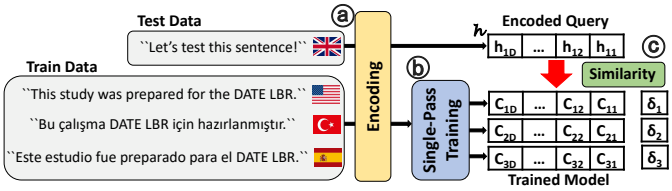

Fig. 1. Overview of an HDC system: encoding, training, and similarity check.

are encoded via $\boldsymbol{P}$s that are orthogonal to each other. On the contrary, level hypervectors are expected to have correlations between neighbors. The final hypervector is denoted as $\boldsymbol{\mathcal{H}} = \Sigma_{i=1}^{N}(\boldsymbol{L}_i \oplus \boldsymbol{P}_i)$, where N is the feature size. The second category utilizes n-gram-based statistics like those in natural language processing systems. These encoders use rotationally permuted hypervectors, which are orthogonal to each other. The final hypervector is $\boldsymbol{\mathcal{H}} = \boldsymbol{L}_1 \oplus \pi\boldsymbol{L}_2 \oplus \pi^{N-1}\boldsymbol{L}_N$, where π^n denotes the n-times rotationally permuted $\boldsymbol{L}$. All samples in the training dataset are evaluated for $\boldsymbol{\mathcal{H}}$, and each contributes to the corresponding class hypervector, which is the *trained model* of the overall system. During the inference, the test data is encoded ($\boldsymbol{h}$), and the similarity check is performed between each test query and the class hypervector. In our encoding scheme with Sobol sequences, we utilize the n-gram-based approach and test the language classification problem [6].

II. Quasi-Random Hypervectors for HDC

Recently, quasi-random Sobol sequences have been used to improve the performance of stochastic computing (SC) systems by generating fast-converging LD bit-streams [2]. Sobol-based bit-stream generators also come with a lightweight hardware design [7]. An analogy from bit-streams of SC to hypervectors of HDC, this work utilizes Sobol sequences in vector encoding of HDC systems. Any readily available Sobol sequence (e.g., in MATLAB via `sobolset()` point set) with a D dimension ($\mathbf{S}_D \in [0,1]$) can be considered an array with pre-allocated quasi-random numbers. For each i^{th} hypervector position, $\mathbf{S}$ is compared with a threshold value (t). If $t > \mathbf{S}_i$, the vector value at the i^{th} position is $+1$; otherwise, it is -1. In conventional HDC systems with pseudo-random *nearly* orthogonal vectors [1], t is generally 0.5. Fig. 2-I-(a) depicts an example of hypervector generation with MATLAB's first Sobol sequence.

After encoding hypervectors, the next step is training (Fig. 2-I-(b)). Considering the letter processing example [6], every n-character block in the training dataset is subject to multiplication (`XOR`), permutation (shifting), and cumulative addition (popcount) over previously encoded hypervectors (uniquely assigned to each letter). The accumulated values (`Acc`) are binarized at each vector position using the `Sign` function. The class hypervector of each language is obtained after a complete scan of the dataset for that class. For test set classification accuracy, the text hypervector obtained for each query is compared with the class hypervector (Fig. 2-I-(c)). This study uses cosine similarity for this comparison.

978-3-9819263-7-8/DATE23/© 2023 EDAA

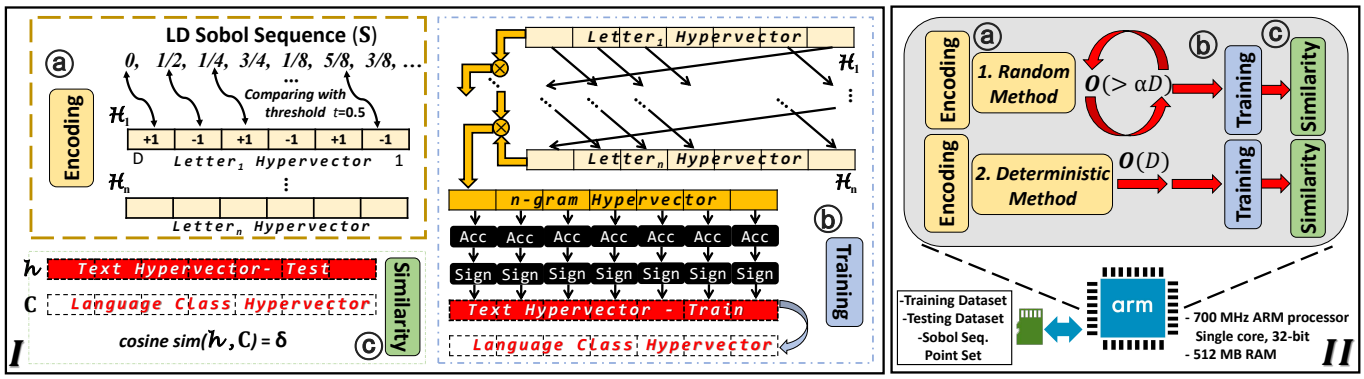

Fig. 2. A new encoding in an n-gram-based HDC system illustrating letter processing: I - New encoding, conventional training, and similarity check in the inference. II - Deployment of the proposed approach.

III. Design Evaluation

Fig. 2-II illustrates deploying an HDC system to an ARM processor for training and inference to prove edge device compatibility. A 700 MHz, 32-bit, single-core ARM processor runs the HDC system implemented in C language. The training and testing data are read from an SD card. We evaluate two encoding methods: *1*) the conventional random approach and *2*) the deterministic approach based on Sobol sequences. For optimization, the random method generates different letter hypervectors in rounds of α times, consuming a runtime of $O(>\alpha D)$ complexity. The random method dynamically creates data with a built-in C language-based `rand` function. On the other hand, the Sobol sequences are pre-generated, stored in, and read from memory.

We evaluate the accuracy and speed of the two approaches. We use the 21-class European languages dataset [8] to train the HDC system and the Europarl Parallel Corpus dataset [9] for the inference. The selected n-gram was *four* decided by iterating over $n \in \{2, 3, 4, 5\}$ for its outperformance. Table I shows the classification accuracy of the two encoding approaches. For the random approach, α is selected as 1:1:1000 iteratively, and the average classification accuracy is reported over $1,000$ independent runs. For this approach, we also report the minimum and maximum accuracy values. As can be seen in Table I, the proposed encoding outperforms the conventional random approach by bringing better-uncorrelated hypervectors for all D sizes. The best accuracy values are achieved with the proposed encoding. Compared to the state-of-the-art HDC language classification [6] ($D = 10,000$, $n = 4$, *accuracy* $= 97.1\%$), the Sobol-based approach improves the classification rate by 0.75% when $D = 8,192$.

Table I also presents the performance (i.e., run-time and memory usage) results. Training on the edge device was performed with hypervectors of length $D = 8,192$. Encoding methods were tested independently. Runtime comparison was performed for single-time hypervector assignment for a fair comparison ($\alpha = 1$ for the random approach and Sobol-based encoding always works with a single iteration). As can be seen, the proposed encoding achieves a better runtime. It should be noted that the random method needs to be run iteratively to achieve comparable accuracy with the proposed encoding. Therefore, even though the random numbers are pre-stored like in Sobol (for better runtime), multiple runs are required for better orthogonality and so higher accuracy. Moreover, if new symbols are added to the HDC system, new hypervector generations need further optimizations. The proposed encoding, however, loads and uses only an additional Sobol sequence, thus providing a dynamic architecture.

IV. Conclusions

This study proposes an encoding method that alleviates the training step in HDC systems. Without utilizing any pre-optimization or exhaustive search for the best-performing hypervectors, Sobol-based encoding yields promising accuracy and performance results. Considering the recent efforts on in-memory computing-based HDC systems, deployment of full training into emerging platforms is challenging. Utilizing a single-read operation for hypervector encoding, as proposed in this work, shows promising results for developing fast and edge-compatible HDC systems.

TABLE I
Accuracy and Performance Results

		Accuracy		
D	**Encoding**	**Minimum**	**Average**	**Maximum**
256	**Random**	67.36%	69.24%	71.70%
	Sobol		**79.03%**	
512	**Random**	82.32%	83.03%	83.83%
	Sobol		**89.47%**	
1,024	**Random**	90.27%	91.15%	91.51%
	Sobol		**93.78%**	
2,048	**Random**	94.73%	95.14%	95.40%
	Sobol		**96.31%**	
4,096	**Random**	96.68%	96.88%	97.09%
	Sobol		**97.05%**	
8,192	**Random**	97.47%	97.68%	97.87%
	Sobol		**97.85%**	

		Performance	
D	**Encoding**	**Runtime**	**Memory**
8,192	**Random**	1,068.3 sec	18.3 KB
	Sobol	687.4 sec	17.8 KB

Acknowledgments

This work was supported in part by National Science Foundation (NSF) grants #2127780 and #2019511, SRC Global Research Collaboration, AIHW and HW Security, Department of the Navy, Office of Naval Research, grant #N00014-21-1-2225 and #N00014-22-1-2067, Air Force Office of Scientific Research, grant #22RT0060, the Louisiana Board of Regents Support Fund #LEQSF(2020-23)-RD-A-26, and generous gifts from Cisco, Xilinx, and Nvidia.

References

[1] P. Kanerva. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive Computation*, 1(2):139–159, 2009.
[2] S. Liu and J. Han. Energy efficient stochastic computing with sobol sequences. In *2017 DATE*, pp. 650–653, 2017.
[3] A. Rahimi *et al.* Efficient biosignal processing using hyperdimensional computing: Network templates for combined learning and classification of exg signals. *Proceedings of the IEEE*, 107(1):123–143, 2019.
[4] L. Ge and K. K. Parhi. Classification using hyperdimensional computing: A review. *IEEE Circ. and Syst. Mag.*, 20(2):30–47, 2020.
[5] Y. Yao *et al.* Fast sar image recognition via hyperdimensional computing using monogenic mapping. *IEEE Geo. and Rem. Sens. Let.*, 19:1–5, 2022.
[6] A. Rahimi *et al.* A robust and energy-efficient classifier using brain-inspired hyperdimensional computing. pp. 64–69, 2016.
[7] M. H. Najafi *et al.* Deterministic methods for stochastic computing using low-discrepancy sequences. In *2018 ICCAD*, pp. 1–8, 2018.
[8] U. Quasthoff *et al.* Corpus portal for search in monolingual corpora. In *LREC*, 2006.
[9] P. Koehn. Europarl. http://www.statmt.org/europarl/, 2005.