No-Multiplication Deterministic Hyperdimensional Encoding for Resource-Constrained Devices

Mehran Shoushtari Moghadam, *Graduate Student Member, IEEE*, Sercan Aygun, *Member, IEEE*, and M. Hassan Najafi, *Member, IEEE*

***Abstract*—Hyperdimensional vector processing is a nascent computing approach that mimics the brain structure and offers lightweight, robust, and efficient hardware solutions for different learning and cognitive tasks. For image recognition and classification, hyperdimensional computing (HDC) utilizes the intensity values of captured images and the positions of image pixels. Traditional HDC systems represent the intensity and positions with binary hypervectors of 1K–10K dimensions. The intensity hypervectors are cross-correlated for closer values and uncorrelated for distant values in the intensity range. The position hypervectors are pseudo-random binary vectors generated iteratively for the best classification performance. In this study, we propose a radically new approach for encoding image data in HDC systems. Position hypervectors are no longer needed by encoding pixel intensities using a deterministic approach based on quasi-random sequences. The proposed approach significantly reduces the number of operations by eliminating the position hypervectors and the multiplication operations in the HDC system. Additionally, we suggest a hybrid technique for generating hypervectors by combining two deterministic sequences, achieving higher classification accuracy. Our experimental results show up to 102× reduction in runtime and significant memory-usage savings with improved accuracy compared to a baseline HDC system with conventional hypervector encoding.**

***Index Terms*—Deterministic bit-streams, hyperdimensional computing (HDC), hypervector encoding, stochastic computing.**

I. Introduction

UNCONVENTIONAL data representations using long *binary sequences* in the form of stochastic bit-streams [1], unary bit-streams [2], and hypervectors [3] have come to the fore of the emerging computing technologies in the last decade. Unlike conventional binary radix representation, these binary sequences assign equal weight to all bits independent of their positions. Such holographic representation provides robustness to soft errors (i.e., bit flips), as any single-bit fault can lead to only a least significant bit error. It also realizes complex computations using simple hardware-friendly operations. Hyperdimensional computing (HDC) is an emerging computing paradigm based on the observation that the human brain operates on high-dimensional data. HDC can transform data into knowledge at a very low cost and with better or comparable accuracy to state-of-the-art (SOTA) methods for diverse learning and cognitive applications [4]. The data are encoded in HDC systems with high-dimensional vectors of -1s (logic-0 in hardware) and $+1$s (logic-1 in hardware). Fig. 1 illustrates the basic steps of HDC.

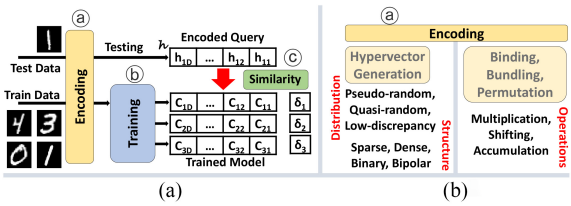

Fig. 1. Basic HDC steps. (a) Training–testing for ML model. (b) Hypervector encoding details.

With an analogy to the traditional machine learning (ML) systems, HDC provides efficiency with ❶ simple arithmetic operations, ❷ noniterative learning, ❸ optimization-less model tuning, and ❹ small model size. In conventional neural networks (NNs), three basic operations are utilized for the forward pass of the learning phase: 1) multiplication; 2) addition; and 3) activation. Similarly, in HDC, hypervectors are 1) multiplied; 2) accumulated; and 3) binarized during the model obtainment. However, HDC only uses simple logical operators: multiplication, also known as binding, with XORs, addition with *population counters*, and binarization with *comparator* (❶). While HDC ends up with a final model after these fundamental operations, conventional ML with NNs needs optimization over an error check using operation-intensive partial derivatives and model updates (❷, ❸). The HDC models consist of binary sequences of -1s and $+1$s, building each class with D-size hypervectors instead of positional weights in matrices linking loads of neurons in the consecutive layers (❹).

For image classification, the current HDC systems encode image data using pixel intensities and the corresponding positions (assuming that the system operates on 2-D grayscale images, as shown in Fig. 1). SOTA studies call this approach *record-based encoding* [5]. Two important parameters—*intensity values* and *pixel positions*—are represented with hypervectors. The record-based encoding assigns level hypervectors, $\boldsymbol{L}$, for the intensity values. Since the pixel intensities are numerical values, closer numbers have similar vector encoding. For D-dimensional vectors, n-bit grayscale intensity representation conforms hypervectors between D-sized $\boldsymbol{L}_0 = 000..00_D$ (full of logic-0s, i.e., -1s) and $\boldsymbol{L}_{2^n} = 111..11_D$ (full of logic-1s, i.e., $+1$s). A random bit-flipping approach is utilized starting from the first level hypervector, $\boldsymbol{L}_0$. At each level, $k = (D/2^n)$ bits are randomly flipped ($0 \leftrightarrow 1$ *flip*). On the other hand, each pixel's position is considered in a 2-D (x, y) image matrix. Since positions are not numerical values, each coordinate is represented with a symbolic assignment of random hypervectors. These are called position

Manuscript received 13 July 2023; accepted 22 July 2023. Date of publication 25 September 2023; date of current version 28 November 2023. This work was supported in part by the National Science Foundation (NSF) under Grant 2019511; by the Louisiana Board of Regents Support Fund under Grant LEQSF(2020-23)-RD-A-26; by Cisco; by Xilinx; and by NVIDIA. This manuscript was recommended for publication by P. R. Panda. *(Corresponding author: Sercan Aygun.)*

The authors are with the School of Computing and Informatics, University of Louisiana at Lafayette, Lafayette, LA 70503 USA (e-mail: mehran.shoushtari-moghadam1@louisiana.edu; sercan.aygun@louisiana.edu; najafi@louisiana.edu).

Digital Object Identifier 10.1109/LES.2023.3298732

1943-0671 © 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: University of Louisiana at Lafayette. Downloaded on April 02,2024 at 20:16:00 UTC from IEEE Xplore. Restrictions apply.

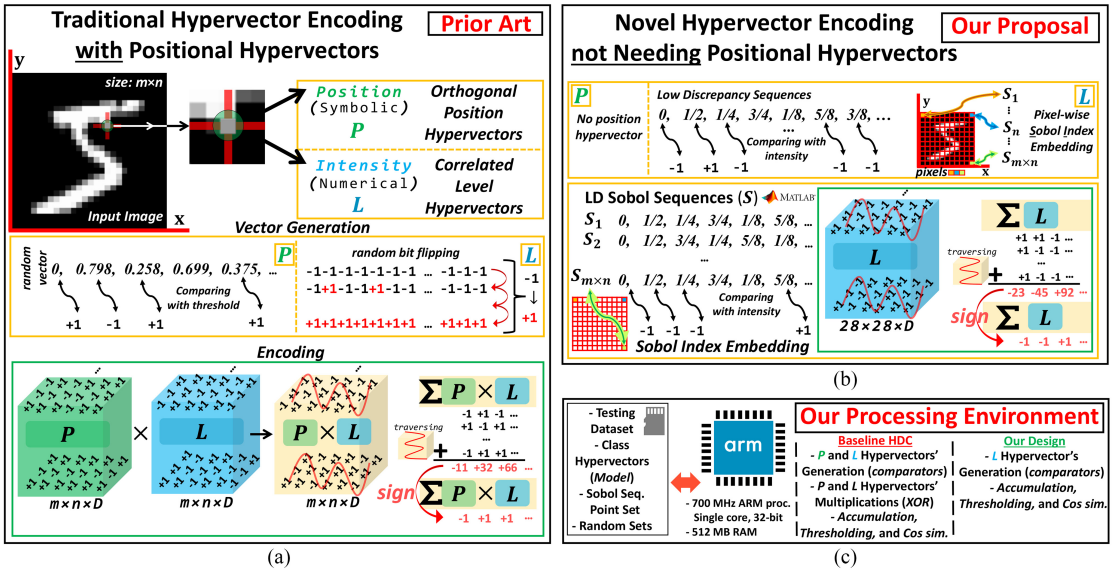

Fig. 2. Image hypervector encoding for cognitive tasks of HDC. (a) Prior art for baseline HDC having conventional position-based encoding. (b) Proposal in this work using LD Sobol sequences that alleviate position-based operations. (c) Our deployment procedure and processing environment.

hypervectors, denoted as $\boldsymbol{P}$. Any pixel position, (x_i, y_j), has a global representation of $\boldsymbol{P}_{i,j}$, valid for training or inference images. $\boldsymbol{P}$s are orthogonal to each other. Random hypervector assignment approximately provides this orthogonality. Each $\boldsymbol{P}$ has 50% logic-1s and 50% logic-0s. After encoding pixel and position hypervectors, the final encoded image is obtained by multiplying and accumulating the values of $\boldsymbol{L}$s and $\boldsymbol{P}$s: $\mathcal{H} = \Sigma_{i=1}^{N}(\boldsymbol{L}_i \oplus \boldsymbol{P}_i)$, where N is the image size (`row` × `column`).

This letter proposes a novel approach for encoding image data in HDC systems. We utilize low-discrepancy (LD) sequences [6], such as Sobol ($\boldsymbol{S}$), Halton, or Van Der Corput (VDC) sequences, for *deterministic* encoding of the image intensity values. We do not encode positions; instead, we use the corresponding index of any sequence (e.g., Sobol $\boldsymbol{S}_i$) ranging from $\boldsymbol{S}_1$ to $\boldsymbol{S}_{\texttt{row×column}}$. Finally, the image hypervector formula becomes $\mathcal{H} = \Sigma_{i=1}^{N}(\boldsymbol{L}_i)$. Our experimental results on an image classification HDC system show that the proposed encoding technique reduces the inference runtime by 102× with 8K vector dimensions for the MNIST dataset. The accuracy of the proposed method surpasses the baseline and SOTA approaches for the CIFAR-10 dataset [7], [8], [9].

II. From Random to Deterministic Encoding

Current HDC systems provide orthogonality by generating pseudo-random hypervectors [10]. The best random vectors are selected over iterations for the best performance. In this letter, we develop a hypervector encoding technique using LD sequences that is *deterministic* and *highly accurate* and does not need any iteration. Generating two distinct hypervector data structures ($\boldsymbol{P}$ and $\boldsymbol{L}$) and applying logical processing (`XOR`, multiplication) on them can be very time and energy-consuming. As an important advantage, our technique operates on a single hypervector data, $\boldsymbol{L}$. We further propose a hybrid technique for generating hypervectors by combining different types of deterministic sequences instead of a single type, providing a higher classification accuracy. Our technique relies on LD sequences, initially employed for stochastic computing (SC), and extends their application to HDC. In the literature, several methods have been proposed to improve the accuracy of HDC models. Yan et al. [11] introduced an efficient HDC approach with lower hypervector dimensions, incorporating retraining to achieve higher accuracy. Similarly, QuantHD [8] quantizes the model and includes a retraining phase to compensate for accuracy loss. Imani et al. [7] proposed SearcHD, which encodes data points into a hyperdimensional space using binary elements and utilizes in-memory computation for training. They assign multiple binary hypervectors to each class. Another approach [9] incorporates a binary NN (BNN) to improve inference accuracy. All these SOTA methods utilize optimization techniques such as NN assistance, retraining, quantization, or learning to enhance accuracy. However, our approach in this study employs a plain HDC without applying any additional optimization technique.

Very few studies have explored the analogy between SC and HDC [12], [13], [14], despite the significant alignment of these emerging computing paradigms. This work is the first to exploit the deterministic techniques from SC for possible image and symbol encoding in HDC.

A. Novel Hypervector Encoding

LD Sobol sequences have proven their effectiveness for SC systems [1], [15], [16]. They are used to generate ideal independent bit-streams for accurate SC operations. In this letter, we use Sobol sequences for dynamic hypervector generation. We compare the normalized intensity values with Sobol numbers from different Sobol sequences to generate high-quality hypervectors.

Fig. 2(a) shows the traditional hypervector encoding for image hypervectors (*baseline HDC*). $\boldsymbol{P}$s are generated by comparing the random numbers with a threshold value of 0.5. This is an unbiased value to produce an equal number of $+1$s and -1s in each hypervector, thereby having better orthogonality. Level hypervector generation is based on bit flipping. The generated hypervectors are then multiplied element-wise (bit-wise `XOR` in hardware). To accumulate the multiplied hypervectors ($\boldsymbol{L} \oplus \boldsymbol{P}$) coming from each pixel, positions are traversed. Hypervectors are added to each other again by element-wise processing (bit-wise popcount in hardware). The final values are evaluated for class hypervector, and binarization is performed via `sign` function (thresholding with comparator in hardware). For each class in training, data are processed to contribute to the corresponding class hypervector. In inference, each query image from the test set is evaluated for the same encoding steps, and

Authorized licensed use limited to: University of Louisiana at Lafayette. Downloaded on April 02,2024 at 20:16:00 UTC from IEEE Xplore. Restrictions apply.

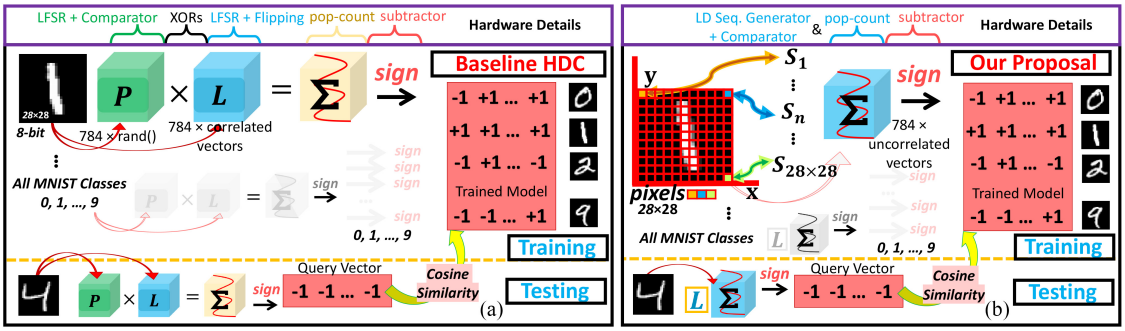

Fig. 3. Example training and testing of Baseline HDC and our proposal. MNIST and CIFAR-10 with 8-bit grayscale images are utilized. (a) Baseline HDC: $\boldsymbol{P}$s are generated using pseudo-random techniques based on the position of each pixel. $\boldsymbol{L}$s are generated using random bit-flipping methods. LFSR is used for $\boldsymbol{P}$ and $\boldsymbol{L}$ generation. $\boldsymbol{P}$s and $\boldsymbol{L}$s are multiplied (binding—via `XORS`), and the results are used for addition (bundling—via pop-counters). Thresholding (or subtraction) is used for the sign function. In software simulations, the `rand()` function is used for $\boldsymbol{P}$ and $\boldsymbol{L}$ generation, and arithmetic operators are used for multiplication, addition, and the `sign()` function. (b) Our proposal: $\boldsymbol{P}$s are not utilized. $\boldsymbol{L}$s are generated using quasi-random methods (LD sequence generators). There is no binding. $\boldsymbol{L}$s are directly used for bundling (via pop-counters). Thresholding (or subtraction) is used for the sign.

the obtained query hypervector is compared with each class' trained hypervectors, e.g., using cosine similarity. The highest similarity is the classification result.

Fig. 2(b) shows our proposed encoding technique. We radically change the encoding in cognitive HDC systems by alleviating their operations. Any pixel intensity is encoded based on the pixel position corresponding to the LD sequence index; this is exemplified in Fig. 2(b) with Sobol sequences. The normalized intensity value (by D) is compared with each element in the corresponding Sobol sequence. If the normalized intensity is greater than the Sobol random number, the hypervector position gets $+1$; otherwise, it gets -1. After obtaining $\boldsymbol{L}$, we perform the accumulation without the multiplication step of the encoding. The rest of the operations are the same as those in the baseline HDC.

B. Hybrid Deterministic Encoding

In addition to Sobol sequences, we explore two high-quality LD random sequences, Halton and VDC, for HDC encoding. The *discrepancy* term in LD refers to the deviation from uniformity within the sample space. Thus, high orthogonality is expected by using LD sequences, and the HDC system can benefit from them. The VDC sequences play a fundamental role and serve as the foundation for some other LD sequences, such as Halton sequences. Generally, each VDC sequence is constructed by reflecting the digit values of a given number with respect to the radix point for a base-B number and then representing the resulting number within the $(0, 1)$ interval. This can be realized efficiently in hardware.

The HDC system may require many random sequences for the encoding stage. For instance, the number of needed random sequences may correspond to the number of image pixels. Due to the limitation in the number of available LD sequences in some tools (e.g., `MATLAB` has 1111 built-in Sobol sequences and Python has 21 201 different sequences), we also explore the case of using different types of sequences (e.g., Sobol + VDC).

III. Design Evaluation

Fig. 3 presents the overall structure of the baseline HDC model [Fig. 3(a)], and the proposed method [Fig. 3(b)] considering the training and testing phases. The baseline HDC is constructed without the assistance of retraining or NN. During the training phase, the $\boldsymbol{P}$ and $\boldsymbol{L}$ hypervectors are generated using pseudo-random methods, e.g., linear-feedback shift register constructs. The results are then multiplied by `XOR` operations and then processed through `pop-counting` operations. The binarized results, obtained through subtraction, are stored as models for each input. The same procedure is applied during the testing phase, where the resulting binary hypervector is compared with the stored hypervectors using `cosine` similarity. In contrast, the proposed method in Fig. 3(b) only utilizes the $\boldsymbol{L}$ hypervectors and does not involve the $\boldsymbol{P}$ hypervectors. We evaluate the accuracy of these models with the MNIST and CIFAR-10 datasets, consisting of 50K images for training and 10K for testing.

TABLE I
Accuracy Comparison for MNIST

Accuracy		Minimum	Average	Maximum
$1K$	**Baseline HDC**	70.65%	79.09%	84.89%
	Our Proposal		83.36%	
$2K$	**Baseline HDC**	71.81%	81.29%	86.96%
	Our Proposal		85.68%	
$8K$	**Baseline HDC**	86.19%	87.27%	87.51%
	Our Proposal		87.60%	

Our proposal: no prior optimizations (e.g., retraining, NN assistance). *SOTA HDCs* (MNIST): ①→ [17] 75.40% (w/o retraining) D=2K | ②→ [18] 86.00% (w/o retraining) D=10K | ③ → [19] 88.00% (w/ retraining) D=10K | ④→ [8], [20] 87.38% (w/ retraining) D=10K.

Performance - Embed.		Run-time	Dyn. Mem.
$1K$	**Baseline HDC**	0.701 sec	8496 KB
	Our proposal	0.016 sec	816 KB
$8K$	**Baseline HDC**	5.938 sec	52 401 KB
	Our proposal	0.058 sec	2220 KB

Code Mem.
Baseline
13.2 KB
Ours
8.2 KB

Fig. 2(c) shows the details of our processing environment. We evaluate the performance of the proposed technique on an embedded platform (700-MHz ARM-based) with limited hardware resources. We first utilize the MNIST dataset for the classification task and compare the performance of the baseline and the proposed encoding technique (when using Sobol sequences) for accuracy, runtime, and memory usage.

Table I compares the accuracy for different dimensions of $1K$, $2K$, and $8K$ with the SOTA HDCs. For the baseline HDC, the training and testing are performed 100 times to report *minimum*, *maximum*, and *average* classification accuracy. Since our approach is free from randomness, it completes in a single iteration. As can be seen, the proposed encoding achieves a higher accuracy than the average and minimum accuracy of the baseline encoding for all dimensions. We also compare the inference runtime per image. We can see that the proposed technique reduces the runtime by 43× for $1K$ and 102× for $8K$. The memory usage is also reduced by 10.4× for $1K$ and 23.6× for $8K$ when using the proposed technique.

To further explore the LD sequences, we utilize hybrid combinations of Sobol + VDC and Halton + VDC sequences. These increase the design space by increasing the indexes derived from merged sequences. We observed that the combined usage of Sobol and VDC sequences results in better outcomes than using the Sobol sequences alone. This enhancement can be attributed

Authorized licensed use limited to: University of Louisiana at Lafayette. Downloaded on April 02,2024 at 20:16:00 UTC from IEEE Xplore. Restrictions apply.

TABLE II
ACCURACY COMPARISON OF HYBRID SEQUENCES USING MNIST

D	$1K$	$2K$	$8K$
Sobol + VDC	85.10%	87.12%	88.68%
Halton + VDC	82.33%	86.48%	88.30%

TABLE III
HARDWARE COST OF TWO HYPERVECTOR GENERATIONS WITH $D = 256$

Seq.	**Area** (μm^2)	**Power** (μW)	**Latency** (ns)
Sobol	1562	24.64	0.68
Halton	580	50.45	1.06
VDC	163	16.05	0.49

TABLE IV
ACCURACY COMPARISON FOR CIFAR-10: SOTA HDCS VERSUS THIS WORK

Method	**Acc. (%)**	**Retrain**	**NN Assist.**
SearcHD ($10K$) [7] [9] [11]	22.66	No	No
BRIC ($4K$) [22] [21]	26.90	Yes	No
QuantHD ($8K$) [8] [9] [11]	28.42	Yes	No
LeHDC ($10K$) (BNN) [9]	46.10	Yes	Yes
Efficient HD (32) (BNN) [11]	46.18	Yes	Yes
GlueHD ($8K$) (Multiple NNs-MNNs) [23]	66.20	Yes	Yes
GlueHD ($8K$) (VGG11 + MNNs) [23]	90.80	Yes	Yes
Baseline HDC ($10K$) [9]	29.50	No	No
*This work-1 ($8K$) (Sobol + VDC)	42.72	No	No
*This work-1 ($8K$) (Halton + VDC)	34.72	No	No
*This work-2 ($8K$) (Sobol) (CNN†)	53.62	No	Yes
*This work-2 ($8K$) (Halton) (CNN†)	52.94	No	Yes
*This work-3 ($8K$) (Sobol) (VGG11)	90.97	Yes	Yes

This work-1: plain-HDC, This work-2: CNN† feature extraction assistance, This work-3: VGG-11 as a pre-trained model for transfer learning. †CNN uses three convolutional layers in this work.

to the perfect interorthogonality between the sequences and their intraorthogonality. Table II presents the accuracy results for the classification of the MNIST dataset.

In addition to evaluating the performance of our encoding technique on an embedded system, we compare the hardware ASIC costs of the proposed hypervector generators. Table III reports the hardware cost of three hypervector generation methods using the Synopsys Design Compiler v2018.06 and the 45-nm FreePDK gate library. The VDC sequence generator design employs a simple $\log_2 D$-bit counter to generate sequence values for a specific base of D, assuming that D is a power of 2. This sequence generator benefits from its simplicity and lightweight design, and as it can be seen, provides approximately 3.5× and 9.58× greater area efficiency compared to the Halton sequence generator and Sobol-based design, respectively. In terms of power consumption, the VDC design is approximately 3.14× and 1.53× more efficient than the Halton and Sobol sequence design, respectively.

To widen the scope of our study and encompass a broader array of applications, we expand our analysis to the CIFAR-10 dataset. We evaluate the performance of the HDC architecture not only in terms of self-accuracy but also in conjunction with NN-assisted systems. Initially, we employed a widely used convolutional NN (CNN) architecture where HDC is used as a classifier. Dutta et al. [21] have previously addressed the potential accuracy degradation of HDC when applied to CIFAR-10 without convolutional features. We measured the performance of self-HDC, HDC functioning as a classifier in the presence of CNN, and HDC with the support of transfer learning, specifically leveraging the VGG11 (Visual Geometry Group, 11-layer depth, pretrained on approximately 1.2 million images from the ImageNet Dataset) architecture. Table IV presents the results compared to the SOTA studies. Notably, the plain HDC, without any NN modifications or retraining, exhibited superior accuracy performance when equipped with our proposed encoding technique. Furthermore, analysis conducted with CNN and transfer learning showed improved accuracy compared to SOTA designs.

IV. CONCLUSION

This study presents a novel encoding method to enhance the accuracy and minimize the runtime and memory usage of hypervector *generation* and *processing* in cognitive HDC systems. We leverage deterministic LD sequences, which offer a fast and precise hypervector encoding approach. Our experimental findings showcase a remarkable 102-fold reduction in runtime and a 23.6-fold decrease in memory usage when employing the proposed technique, compared to the conventional HDC encoding, particularly in embedded environments. Furthermore, we explored utilizing various deterministic sequences and evaluated their impact on accuracy and low-level hardware performance.

REFERENCES

[1] M. H. Najafi, D. J. Lilja, and M. Riedel, "Deterministic methods for stochastic computing using low-discrepancy sequences," in *Proc. ICCAD*, 2018, pp. 1–8.

[2] M. H. Najafi, D. J. Lilja, M. D. Riedel, and K. Bazargan, "Low-cost sorting network circuits using unary processing," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 8, pp. 1471–1480, Aug. 2018.

[3] M. Imani et al., "A framework for collaborative learning in secure high-dimensional space," in *Proc. CLOUD*, 2019, pp. 435–446.

[4] A. Rahimi, P. Kanerva, and J. M. Rabaey, "A robust and energy-efficient classifier using brain-inspired hyperdimensional computing," in *Proc. ISLPED*, 2016, pp. 64–69.

[5] L. Ge and K. K. Parhi, "Classification using hyperdimensional computing: A review," *IEEE Circuits Syst. Mag.*, vol. 20, no. 2, pp. 30–47, 2nd Quart., 2020.

[6] H. Niederreiter, *Random Number Generation and Quasi-Monte Carlo Methods*. Philadelphia, PA, USA: SIAM, 1992.

[7] M. Imani et al., "SearcHD: A memory-centric hyperdimensional computing with stochastic training," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 10, pp. 2422–2433, Oct. 2020.

[8] M. Imani et al., "QuantHD: A quantization framework for hyperdimensional computig," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 10, pp. 2268–2278, Oct. 2020.

[9] S. Duan, Y. Liu, S. Ren, and X. Xu, "LeHDC: Learning-based hyperdimensional computing classifier," in *Proc. DAC*, 2022, pp. 1111–1116.

[10] H. Amrouch et al., "Brain-inspired hyperdimensional computing for ultra-efficient edge AI," in *Proc. CODES+ISSS*, 2022, pp. 25–34.

[11] Z. Yan, S. Wang, K. Tang, and W.-F. Wong, "Efficient hyperdimensional computing," 2023, *arXiv:2301.1090*.

[12] Y. Hao, S. Gupta, J. Morris, B. Khaleghi, B. Aksanli, and T. Rosing, "Stochastic-HD: Leveraging stochastic computing on hyper-dimensional computing," in *Proc. ICCD*, 2021, pp. 321–325.

[13] P. Poduval, Z. Zou, H. Najafi, H. Homayoun, and M. Imani, "StocHD: Stochastic hyperdimensional system for efficient and robust learning from raw data," in *Proc. DAC*, 2021, pp. 1195–1200.

[14] S. Aygun, M. H. Najafi, and M. Imani, "A linear-time, optimization-free, and edge device-compatible hypervector encoding," in *Proc. DATE*, 2023, pp. 1–2.

[15] S. Liu and J. Han, "Energy efficient stochastic computing with Sobol sequences," in *Proc. DATE*, 2017, pp. 650–653.

[16] M. H. Najafi, D. Jenson, D. J. Lilja, and M. D. Riedel, "Performing stochastic computation deterministically," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 12, pp. 2925–2938, Dec. 2019.

[17] S. Datta, R. A. G. Antonio, A. R. S. Ison, and J. M. Rabaey, "A programmable hyper-dimensional processor architecture for human-centric IoT," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 3, pp. 439–452, Sep. 2019.

[18] E. Hassan, Y. Halawani, B. Mohammad, and H. Saleh, "Hyper-dimensional computing challenges and opportunities for AI applications," *IEEE Access*, vol. 10, pp. 97651–97664, 2022.

[19] C.-Y. Hsieh, Y.-C. Chuang, and A.-Y. A. Wu, "FL-HDC: Hyperdimensional computing design for the application of federated learning," in *Proc. AICAS*, 2021, pp. 1–5.

[20] S. Duan, X. Xu, and S. Ren, "A brain-inspired low-dimensional computing classifier for inference on tiny devices," 2022, *arXiv:2203.04894*.

[21] A. Dutta, S. Gupta, B. Khaleghi, R. Chandrasekaran, W. Xu, and T. Rosing, "HDnn-PIM: Efficient in memory design of hyperdimensional computing with feature extraction," in *Proc. GLSVLSI*, 2022, pp. 281–286.

[22] M. Imani, J. Morris, J. Messerly, H. Shu, Y. Deng, and T. Rosing, "BRIC: Locality-based encoding for energy-efficient brain-inspired hyperdimensional computing," in *Proc. ACM/IEEE DAC*, 2019, pp. 1–6.

[23] P. Sutor, D. Yuan, D. Summers-Stay, C. Fermuller, and Y. Aloimonos, "Gluing neural networks symbolically through hyperdimensional computing," 2022, *arXiv:2205.1553*.

Authorized licensed use limited to: University of Louisiana at Lafayette. Downloaded on April 02,2024 at 20:16:00 UTC from IEEE Xplore. Restrictions apply.