# DynHD: Hyperdimensional Computing Approach for Efficient Radar Spectrum Classification

Zhuowen Zou, *Student Member, IEEE*, Yang Ni, *Student Member, IEEE*, SungHeon Jeong, Satish Ravindran, Binbin Shi, Phil Chen, and Mohsen Imani, *Member, IEEE*

***Abstract*—Radar technology plays a critical role in target detection, classification, and tracking. However, the computational demands of training deep neural networks (DNNs) on radar signals can be overwhelming, posing challenges for edge devices with limited energy and computing resources. In this article, we propose leveraging hyperdimensional computing (HDC), a brain-inspired computing paradigm, as an efficient alternative. HDC utilizes high-dimensional vectors for information representation and processing, offering robustness and energy efficiency. We propose a novel HDC classification algorithm named DynHD, with a dynamic HDC encoder that adapts to more challenging radar spectrum recognition tasks. We designed this mechanism to provide great flexibility to the HDC encoder that is otherwise fixed. Our evaluations demonstrate that HDC-based approaches achieve comparable accuracy to DNN-based methods with lower-computational complexity, making them suitable for resource-constrained devices. We achieve significant improvements in latency during training and inference phases, enabling efficient processing of radar signals on edge devices.**

***Index Terms*—Brain-inspired computing, hyperdimensional computing (HDC), radar spectrum recognition.**

## I. Introduction

RADAR technology has become an essential tool for navigation, surveillance, and communication. With radar signals as input, neural networks can learn models to perform target detection, classification, and tracking [1], [2], [3]. However, learning a deep neural network (DNN) is computationally intensive and requires high-performance computing resources.

To address this issue, we turn to brain-inspired hyperdimensional computing (HDC) as a promising alternative framework for the efficient learning of radar data. HDC relies on high-dimensional vectors to represent and process information [4]. It has shown significant potential in various applications for its robustness and energy efficiency [5], [6]. However, the complexity of radar data and the limited feature extraction ability from traditional HDC approaches limit the learning ability [7].

In this article, we propose DynHD, an HDC classification model with dynamic encoding, and compare several HDC-based approaches to perform radar-related classification tasks over edge devices. Our experimental results show that HDC-based approaches can achieve accuracy similar to or better than traditional methods while using significantly fewer resources and fewer training iterations. Our approach enables the processing of radar signals on edge devices with lower latency, making it suitable for real-world applications. In particular, DynHD with 1000 dimensions can achieve around 8× faster training while providing a comparable classification accuracy, compared to the DNN-based counterpart. By comparing different HDC-based methods, we also show that DynHD with an adjustable encoder can provide up to 6× and 100× latency improvement during the training and inference phases. Furthermore, we show that the HDC-based methods are more robust to model quantization and noise from different sources.

Received 24 August 2024; revised 12 October 2024; accepted 21 October 2024. Date of publication 24 October 2024; date of current version 18 April 2025. This work was supported in part by the DARPA Young Faculty Award; in part by the National Science Foundation under Grant 2127780, Grant 2319198, Grant 2321840, Grant 2312517, and Grant 2235472; in part by the Semiconductor Research Corporation (SRC); in part by the Office of Naval Research through the Young Investigator Program Award under Grant N00014-21-1-2225 and Grant N00014-22-1-2067; and in part by the Air Force Office of Scientific Research under Grant FA9550-22-1-0253. This manuscript was recommended for publication by J. Lee. *(Corresponding author: Mohsen Imani.)*

Zhuowen Zou, Yang Ni, SungHeon Jeong, and Mohsen Imani are with the Department of Computer Science, University of California Irvine, Irvine, CA 92697 USA (e-mail: zhuowez1@uci.edu; yni3@uci.edu; sungheoj@uci.edu; m.imani@uci.edu).

Satish Ravindran, Binbin Shi, and Phil Chen are with the Advanced Radar Solutions Department, NXP Semiconductors, San Jose, CA 95134 USA (e-mail: satish.ravindran@nxp.com; binbin.shi@nxp.com; phil.chen_1@nxp.com).

Digital Object Identifier 10.1109/LES.2024.3485638

## II. Related Works

*Radar-Based Target Recognition:* Radar provides all-weather capability, light independence, and velocity measurement, which are crucial for advanced driver-assistance systems (ADAs) [1], [2]. In contrast to LiDAR, radar is more robust against adverse weather conditions and provides excellent cost effectiveness [8]. In prior works, researchers leveraged radar signals to capture stationary and moving targets [9], [10], [11]. However, previous works are mainly based on DNNs, leading to high-resource usage, power consumption, and latency.

*HDC for Machine Learning:* HDC offers significant advantages in various classification and recognition tasks, including text classification [12], human activity recognition [7], [13], and bio-signal processing [14], [15]. Prior works also propose HDC-based algorithms for genomic sequencing [16], [17], reinforcement learning [18], [19], [20], and graph reasoning [21].

## III. Hyperdimensional Computing

HDC leverages high-dimensional vectors called hypervectors as representations, which inherit two important properties [4]: 1) "Hyperdimensionality" and 2) "Holographic Representation."

### A. HDC Operations

Four operators form the building blocks of HDC algorithms: bundling $\oplus$, binding $\odot$, permutation $\rho$, and the similarity

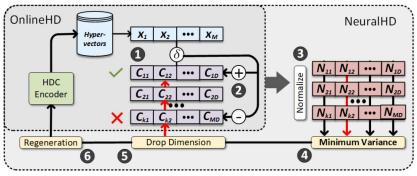


Fig. 1. OnlineHD and NeuralHD training.

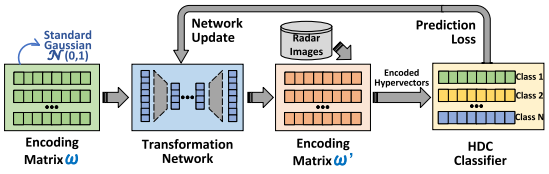


Fig. 2. Overview of the DynHD architecture and learning process.

function $\delta$. Let $\mathbb{X}$ be the space of variables. For any three variables $x, y, z \in \mathbb{X}$, we assign to each an independently and randomly sampled hypervector with dimensionality $D$ in the range of 1k to 10k. Each component of the hypervector $H_x$, $H_y$, and $H_z$ is $-1$ or 1 with equal probability. *Similarity* $\delta$ is the normalized dot-product, where the normalization is of $l_0$ norm: $\delta(H, H') = (1/D)H^T H'$. The similarity between different hypervectors $\delta(H_x, H_y) \approx 0$, while the similarity between one and itself is $\delta(H_x, H_x) = 1 \gg 0$. *Bundling* creates a set. For the set $S = \{x, y\} \in X$, $H_S = H_x \oplus H_y$, where $\oplus$ is the element-wise addition. As a result, $\delta(H_S, H_x) = \delta(H_x, H_x) + \delta(H_y, H_x) \gg 0$ while $\delta(H_s, H_z) \approx 0$. *Binding* represents association. To associate $x$ with $y$, $H_{(x,y)} = H_x \odot H_y$, where $\odot$ is the element-wise multiplication. The resulting vector is dissimilar to its constituents, $\delta(H_{(x,y)}, H_x) \approx x$. *Permutation* over a hypervector creates a dissimilar vector and thus implements order and indexing. In particular, we may represent "$y$ is in the $i^{th}$ slot" as $H_y^{(i)} = \rho^i H_y$ for some fixed random permutation $\rho$ [22]. For $i \neq j$, $\delta(H_y^{(i)}, H_y^{(j)}) \approx 0$. In practice, the permutation is implemented by a circular shift.

### B. Traditional HDC Encoding and Learning

*Record-Based Encoder [23]:* For input data of $d$ dimensions, the model generates an address codebook with one entry for each dimension $\{A_1, \ldots, A_d\}$. And a value codebook $\{L_1, \ldots, L_q\}$ corresponding to the $q$ quantized levels of the continuous value. These codebooks contain hypervectors of dimensionality $D$. The encoder then performs a bundling of the address-value pair association for a data point $x \in \mathbb{R}^d$: $H_x = \sum_{i=1}^{d} A_i \odot V_i$, $V_i \in \{L_1, \ldots, L_q\}$.

*Ngram-Based Encoder [24]:* For a feature value in each position, the model permutes the corresponding value hypervector according to the position of the features: $H_x = \sum_{i=1}^{d} \rho^i(V_i)$, $V_i \in \{L_1, \ldots, L_q\}$. *Learning:* An HDC model will generate a class hypervector for each class by bundling hypervectors of the data of the class. The inference of a query hypervector is then done by finding the class hypervector with the highest similarity.

## IV. HDC Model Design

We first implement two more recent HDC-based classification algorithms (OnlineHD [6] and NeuralHD [5], as shown in Fig. 1) and then propose DynHD that significantly boosts the learning quality, all of which leverages nonlinear HDC encoding for a better representation. In summary, we first introduce the nonlinear HDC encoding used by all three models. Then, we introduce OnlineHD as a baseline that employs a static encoder and an adaptive training scheme. NeurlaHD extends OnlineHD by incorporating dynamic hyperdimension regeneration, which can be considered an evolutionary approach to improving the encoding matrix. Finally, we introduce our approach, DynHD, which takes a step further with a more general form of dynamical update.

*Nonlinear HDC Encoding:* Our HDC encoder leverages kernel methods to maintain the property of HDC representation while preserving the important structure of the radar data. We encode radar spectrum through their random Fourier features (RFFs) under the radial basis function (RBF) kernel [25]. This implies a $D$-dimensional HDC encoding, where each component is a RFF: $H_x = [\zeta_{\omega_1}(x), \ldots, \zeta_{\omega_D}(x)]^T$, $\omega_i \sim p, i = 1, \ldots, D$, where $\zeta_\omega(x) = e^{i\omega' x}$ and $p$ is a probability measure. To avoid storing and computing complex numbers, there exists an equivalent encoding method. For input $x \in \mathbb{R}^d$: $H_x = \cos(\Omega x + b)$, where $\Omega \in \mathbb{R}^{D \times d}$ is a random matrix with each entry sampled from $N(0, 1)$ and $b \in \mathbb{R}^D$ is a random vector with each entry sampled from $U[0, 2\pi]$. When coupled with the normalized dot product as the similarity function (Section III-A), the hypervector similarity approximates the RBF kernel [25]: $\delta(H_x, H_y) \approx \exp(-[\|x - y\|_2^2]/2)$.

*OnlineHD—HDC With Adaptive and Online Learning:* It [6] is an adaptive training framework for efficient and accurate HDC learning. Fig. 1 shows the adaptive training routine. Instead of naively bundling hypervectors of the same class, OnlineHD adds and subtracts hypervectors in a weighted manner. For a training data point $H$, OnlineHD evaluates its similarity with all class hypervectors $C_i$ (❶), and updates the model as follows (❷): $\mathcal{C}_l \leftarrow \mathcal{C}_l + \eta(1 - \delta_l) \times \mathcal{H}$ and $\mathcal{C}_{l'} \leftarrow \mathcal{C}_{l'} - \eta(1 - \delta_{l'}) \times \mathcal{H}$, where $l$ is the correct class and $l'$ the predicted class, $\eta$ is a learning rate.

*NeuralHD—Dynamic Hyperdimension Regeneration:* It [5] includes a mechanism to dynamically update the encoding. NeuralHD evaluates the significance of hyperdimension, using normalized variance (across the classes) as the measure (❸, ❹). NeuralHD then replaces these dimensions in the encoders and classifiers by (1) generating new values for corresponding rows in the encoding codebook $\Omega$ (❺), and (2) optionally zero-out the corresponding dimensions in the class hypervectors (❻). The dynamic encoder allows hyperdimension to be used effectively and improves accuracy.

*DynHD—Dynamic, Adaptive, and Trainable Hypervector Encoding:* To meaningfully improve the accuracy of the model, we leverage kernel theory to enable the specialization of the encoder for radar tasks. In DynHD, we perform a more general form dynamical update of the encoder using a learned transformation to the randomly generated encoding matrix. Fig. 2 presents the overview of DynHD. We leverage a differentiable encoder-decoder structure, where the inputs are randomly generated hypervectors in the encoding matrix and the outputs are transformed hypervectors of the same dimensionality. This network has two fully connected layers that map the input dimensionality from $D \rightarrow 256 \rightarrow D$. The encoding matrix is then used to encode data into hypervectors. After every few batches, the transformation network is updated according to the loss collected by the HDC classifier. The backpropagation flows through the HDC classifier and the encoding matrix $\omega'$, and directly guides the update of the transformation network. The encoding matrix $\omega'$ and the classifier

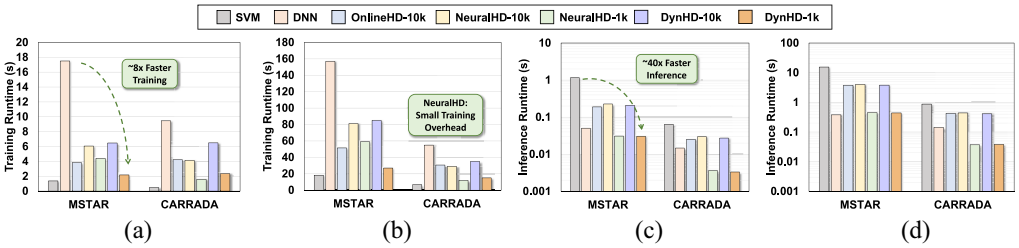


Fig. 3. Efficiency comparison on different radar datasets: (a) and (b) Training runtime on laptop and embedded CPU and (c) and (d) inference runtime on both CPUs.

remain frozen during the update. It is enough to update the transformation network every ten batches of HDC classifier training (i.e., two types of training interleaves). Once the update is performed, we resample the random matrix and input it to the transformation network to generate the new encoding matrix. The process repeats until the model converges. The key benefit of the approach is that DynHD encoding does not require an explicit form of the kernel function, the probability density, or expensive Fourier transforms. Because of Bochner's theorem, learning the transformation is equivalent to learning a subset of continuous shift-invariant kernels, and this method thus provides us with a much richer family of distribution from which we may encode the data, enabling the encoder to specialize to radar data.

## V. Evaluation

### A. Experimental Setup

For hardware platforms, we select a laptop CPU (Intel i7-12700H with 45W TDP) and a low-power embedded CPU on Raspberry Pi (with 6W TDP). We verify DynHD on MSTAR public release database [3] and the CARRADA automotive radar dataset [11]. The former is captured by a 10-GHz synthetic aperture radar (SAR), and the latter is based on the frequency-modulated continuous wave (FMCW) radar. There are ten classes in the MSTAR dataset and 3 classes in the CARRADA dataset. We select several traditional ML methods as a baseline, e.g., decision tree (DT) and support vector machine (SVM). For the deep learning (DL) baseline, we use a DNN with three hidden layers with 512, 256, and 128 neurons, respectively.

### B. Accuracy Comparison With Baseline Methods

Table I includes the inference performance comparison of our models with several widely applied ML algorithms and traditional HDC approaches. Our evaluation shows that HDC-based solutions achieve a notably higher accuracy than SVM and DT while comparable to DNN. DynHD with a trainable HDC encoder achieves better-learning quality than other lightweight baselines. For example, DynHD with 4k dimensionality provides more than 4% higher accuracy than DNN for the MSTAR dataset. Our evaluation also shows that, leveraging dynamic encoding and adaptive training, DynHD significantly outperforms traditional HDC encoding methods. Comparing with OnlineHD validates the effectiveness of the dynamic update in DynHD, and comparing with OnlineHD validates its expressiveness and generality, as DynHD can generate encoding matrices that achieve better performance.

### C. Efficiency Analysis in Training and Inference

Fig. 3 compares the training and inference runtime of our models with other approaches. We record the training time for each algorithm to reach convergence in terms of training accuracy to ensure a fair comparison both in terms of runtime and learning quality. As shown in Fig. 3(a) and (b), compared to DNN, HDC-based solutions, such as OnlineHD-10k and NeuralHD-1k, show significantly faster training. As presented in Fig. 3(c) and (d), the efficiency of NeuralHD-1k and DynHD-1k outperforms other algorithms on average, including DNN and SVM. Table II presents the training power and energy consumption of different algorithms. Table III shows that traditional HDC classification algorithms, including NgramHD and RecordHD, not only achieve lower accuracy but also fail in efficiency. In comparison, DynHD uses a CPU-friendly encoder design and improves the training and inference efficiency by about 6× and 100×, respectively. By introducing DynHD, we can further improve the learning quality at the dimensionality of either 1k or 10k with a small overhead in training while maintaining fast inference. The better efficiency of HDC comes from faster convergence in training and fewer weights to update via backpropagation. Lightweight updates without backpropagation are more suitable for embedded devices.

TABLE I
Accuracy Comparison Across Different ML Models

| Algorithm | MSTAR Accuracy | MSTAR Average Precision | MSTAR Average Recall | MSTAR F1 | CARRADA Accuracy | CARRADA Average Precision | CARRADA Average Recall | CARRADA F1 |
|---|---|---|---|---|---|---|---|---|
| **DT** | 0.579 | 0.425 | 0.446 | 0.432 | 0.673 | 0.587 | 0.588 | 0.587 |
| **SVM** | 0.868 | 0.842 | 0.796 | 0.793 | 0.738 | 0.657 | 0.636 | 0.634 |
| **DNN** | 0.885 | 0.849 | 0.834 | 0.828 | 0.773 | 0.718 | 0.711 | 0.715 |
| **NgramHD-10k** | 0.758 | 0.674 | 0.692 | 0.667 | 0.723 | 0.640 | 0.638 | 0.638 |
| **RecordHD-10k** | 0.709 | 0.609 | 0.636 | 0.608 | 0.732 | 0.631 | 0.627 | 0.629 |
| **OnlineHD-1k** | 0.749 | 0.681 | 0.656 | 0.662 | 0.736 | 0.674 | 0.672 | 0.672 |
| **OnlineHD-4k** | 0.866 | 0.842 | 0.796 | 0.813 | 0.762 | 0.702 | 0.694 | 0.697 |
| **OnlineHD-10k** | 0.902 | 0.889 | 0.847 | 0.861 | 0.762 | 0.699 | 0.691 | 0.694 |
| **NeuralHD-1k** | 0.763 | 0.693 | 0.671 | 0.675 | 0.747 | 0.682 | 0.674 | 0.673 |
| **NeuralHD-4k** | 0.867 | 0.869 | 0.867 | 0.865 | 0.762 | 0.767 | 0.762 | 0.760 |
| **NeuralHD-10k** | 0.885 | 0.865 | 0.822 | 0.836 | **0.778** | 0.720 | 0.711 | 0.715 |
| **DynHD-1k** | 0.911 | 0.913 | 0.911 | 0.911 | 0.675 | 0.677 | 0.675 | 0.675 |
| **DynHD-4k** | **0.926** | **0.928** | **0.926** | **0.926** | 0.739 | 0.736 | 0.739 | 0.734 |
| **DynHD-10k** | 0.921 | 0.924 | 0.921 | 0.920 | 0.741 | **0.743** | **0.741** | **0.740** |

TABLE II
Training Power and Energy Comparison Across Different ML Models on CARRADA

| Algorithms | SVM | DNN | OnlineHD-10k | NeuralHD-10k | NeuralHD-1k | DynHD-10k | DynHD-1k |
|---|---|---|---|---|---|---|---|
| Power (w) | 4.6 | 5.4 | 5.0 | 5.0 | 4.9 | 5.0 | 4.9 |
| Energy (J) | 30.9 | 296.6 | 153.4 | 145 | 58.8 | 175 | 73.5 |

TABLE III
Learning Performance and Efficiency Comparison

| Algorithm | Training Runtime (s) | Inference Runtime (s) | F1 on CARRADA |
|---|---|---|---|
| **NgramHD-10k** | 38.723 | 2.827 | 0.638 |
| **RecordHD-10k** | 37.542 | 2.484 | 0.629 |
| **OnlineHD-10k** | 4.215 | 0.025 | 0.694 |
| **OnlineHD-1k** | 1.216 | 0.003 | 0.672 |
| **NeuralHD-10k** | 4.134 | 0.030 | 0.715 |
| **NeuralHD-1k** | 1.579 | 0.004 | 0.673 |
| **DynHD-10k** | 6.493 | 0.026 | 0.740 |
| **DynHD-1k** | 2.341 | 0.003 | 0.675 |

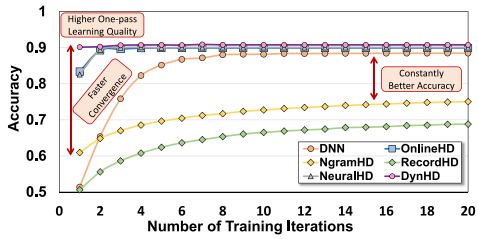


Fig. 4. Learning curve comparison across different algorithms for MSTAR. The accuracy is recorded at the end of each training epoch using the testing dataset. All HDC-based algorithms have dimensionality $D = 10$k.

TABLE IV
LEARNING ACCURACY COMPARISON WITH DIFFERENT MODEL PRECISION

| Models | Datasets | HDC Dim. | Accuracy with Different Model Precision | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | 1-bit Integer | 2-bit Integer | 3-bit Integer | 4-bit Integer | 8-bit Integer | 32-bit Float |
| DynHD | MSTAR | **1000** | 0.842 | 0.857 | 0.863 | 0.862 | 0.859 | 0.911 |
| | | **4000** | **0.862** | **0.867** | **0.868** | **0.868** | 0.865 | **0.926** |
| | | **10000** | 0.854 | 0.857 | 0.856 | 0.857 | 0.855 | 0.921 |
| | CARRADA | **1000** | 0.589 | 0.572 | 0.599 | 0.639 | 0.619 | 0.675 |
| | | **4000** | 0.569 | 0.573 | 0.541 | 0.544 | 0.619 | 0.734 |
| | | **10000** | 0.607 | 0.651 | 0.638 | 0.633 | 0.635 | 0.740 |
| DNN | MSTAR | - | 0.051 | 0.065 | 0.371 | 0.856 | **0.889** | 0.894 |
| | CARRADA | - | 0.206 | 0.256 | 0.643 | 0.704 | 0.740 | 0.744 |

TABLE V
COMPARISON OF MODEL SIZE AND COMPUTATIONAL COST

| Models | Datasets | #MAC | 1-bit Model Size (Byte) | 2-bit | 3-bit | 4-bit | 8-bit | 32-bit |
|---|---|---|---|---|---|---|---|---|
| DynHD | **MSTAR** | 2092000 | 261000 | 522000 | 783000 | 1044000 | 2088000 | 8352000 |
| | **CARRADA** | 1068000 | 129500 | 259000 | 388500 | 518000 | 1036000 | 4144000 |
| DNN | MSTAR | 2163968 | 270496 | 540992 | 811488 | 1081984 | 2163968 | 8655872 |
| | CARRADA | 1639680 | 204960 | 409920 | 614880 | 819840 | 1639680 | 6558720 |

One-pass or few-pass training significantly reduces the learning runtime at the cost of quality. The results in Fig. 4 show that DNN only achieves about 50% accuracy with one-pass training. On the other hand, DynHD shows a notable improvement in terms of one-pass learning accuracy.

In Table IV, we report classification accuracy under different model quantization levels for two radar datasets. We use post-training quantization for both DNN and HDC models. For int8 quantization, we use standard Pytorch quantization; for lower precisions, we use our customized quantization framework based on Pytorch. DynHD is notably more robust to DNN when quantization becomes more aggressive to less than or equal to 4-bit integer precision. We also report the size of the model and the computational cost in Table V.

## VI. CONCLUSION

This work demonstrates the efficacy of HDC in radar spectrum classification. We propose a dynamic, adaptive, and trainable encoding method. Our HDC-based radar recognition algorithm provides comparable or better quality with significant speedups in both learning and inference.

## REFERENCES

[1] S. Chadwick, W. Maddern, and P. Newman, "Distant vehicle detection using radar and vision," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, 2019, pp. 8311–8317.

[2] K. Qian, S. Zhu, X. Zhang, and L. E. Li, "Robust multimodal vehicle detection in foggy weather using complementary lidar and radar signals," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 444–453.

[3] J. R. Diemunsch and J. Wissinger, "Moving and stationary target acquisition and recognition (MSTAR) model-based automatic target recognition: Search technology for a robust ATR," in *Proc. 5th SPIE Algorithms Synth. Aperture Radar Imagery*, 1998, pp. 481–492.

[4] P. Kanerva, "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors," *Cogn. Comput.*, vol. 1, pp. 139–159, Jan. 2009.

[5] Z. Zou, Y. Kim, F. Imani, H. Alimohamadi, R. Cammarota, and M. Imani, "Scalable edge-based hyperdimensional learning system with brain-like neural adaptation," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, 2021, pp. 1–15.

[6] A. Hernández-Cano, N. Matsumoto, E. Ping, and M. Imani, "Onlinehd: Robust, efficient, and single-pass online learning using hyperdimensional system," in *Proc. Design, Autom. Test Europe Conf. Exhibit. (DATE)*, 2021, pp. 56–61.

[7] M. Imani et al., "Revisiting hyperdimensional learning for FPGA and low-power architectures," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit. (HPCA)*, 2021, pp. 221–234.

[8] I. Bilik, "Comparative analysis of radar and lidar technologies for automotive applications," *IEEE Intell. Transp. Syst. Mag.*, vol. 15, no. 1, pp. 244–269, Jan./Feb. 2023.

[9] Z. Yue et al., "A novel semi-supervised convolutional neural network method for synthetic aperture radar image recognition," *Cogn. Comput.*, vol. 13, pp. 795–806, Jul. 2021.

[10] H. Zhu, "Ship classification based on sidelobe elimination of SAR images supervised by visual model," in *Proc. IEEE Radar Conf. (RadarConf)*, 2021, pp. 1–6.

[11] A. Ouaknine, A. Newson, J. Rebut, F. Tupin, and P. Pérez, "Carrada dataset: Camera and automotive radar with range-angle-Doppler annotations," in *Proc. 25th Int. Conf. Pattern Recognit. (ICPR)*, 2021, pp. 5068–5075.

[12] K. Shridhar H. Jain, A. Agarwal, and D. Kleyko, "End to end binarized neural networks for text classification," in *Proc. SustaiNLP, Workshop Simple Effic. Nat. Lang. Process.*, 2020, pp. 29–34.

[13] Y. Ni, Y. Kim, T. Rosing, and M. Imani, "Algorithm-hardware co-design for efficient brain-inspired hyperdimensional learning on edge," in *Proc. Design, Autom. Test Europe Conf. Exhibit. (DATE)*, 2022, pp. 292–297.

[14] Y. Ni, N. Lesica, F.-G. Zeng, and M. Imani, "Neurally-inspired hyperdimensional classification for efficient and robust biosignal processing," in *Proc. 41st IEEE/ACM Int. Conf. Comput.-Aided Design*, 2022, pp. 1–9.

[15] U. Pale, T. Teijeiro, and D. Atienza, "Exploration of hyperdimensional computing strategies for enhanced learning on epileptic seizure detection," in *Proc. 44th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, 2022, pp. 4076–4082.

[16] Z. Zou et al., "BioHD: An efficient genome sequence search platform using hyperdimensional memorization," in *Proc. 49th Annu. Int. Symp. Comput. Archit.*, 2022, pp. 656–669.

[17] H. E. Barkam et al., "HDGIM: Hyperdimensional genome sequence matching on unreliable highly scaled FeFET," in *Proc. Design, Autom. Test Europe Conf. Exhibit. (DATE)*, 2023, pp. 1–6.

[18] Y. Ni, M. Issa, D. Abraham, M. Imani, X. Yin, and M. Imani, "HDPG: Hyperdimensional policy-based reinforcement learning for continuous control," in *Proc. 59th ACM/IEEE Design Autom. Conf.*, 2022, pp. 1141–1146.

[19] Y. Ni, D. Abraham, M. Issa, Y. Kim, P. Mercati, and M. Imani, "Efficient off-policy reinforcement learning via brain-inspired computing," in *Proc. Great Lakes Symp. VLSI*, 2023, pp. 449–453.

[20] H. Chen, M. Issa, Y. Ni, and M. Imani, "DARL: Distributed reconfigurable accelerator for hyperdimensional reinforcement learning," in *Proc. 41st IEEE/ACM Int. Conf. Comput.-Aided Design*, 2022, pp. 1–9.

[21] P. Poduval et al., "Graphd: Graph-based hyperdimensional memorization for brain-like cognitive learning," *Front. Neurosci.*, vol. 16, Feb. 2022, Art. no. 757125.

[22] D. Kleyko, D. Rachkovskij, E. Osipov, and A. Rahimi, "A survey on hyperdimensional computing aka vector symbolic architectures, part II: Applications, cognitive models, and challenges," *ACM Comput. Surv.*, vol. 55, no. 9, pp. 1–52, 2023.

[23] S. Aygun, M. S. Moghadam, M. H. Najafi, and M. Imani, "Learning from hypervectors: A survey on hypervector encoding," 2023, *arXiv:2308.00685*.

[24] S. Moon, B.-T. Berster, H. Xu, and T. Cohen, "Word sense disambiguation of clinical abbreviations with hyperdimensional computing," in *Proc. AMIA Annu. Symp.*, Nov. 2013, pp. 1007–1016.

[25] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Proc. 20th Int. Conf. Neural Inf. Process. Syst.*, 2007, pp. 1177–1184.