2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

# Brain-Inspired Hyperdimensional Computing in the Wild: Lightweight Symbolic Learning for Sensorimotor Controls of Wheeled Robots

Hyukjun Kwon[1], Kangwon Kim[1], Junyoung Lee[2], Hyunsei Lee[2], Jiseung Kim[2], Jinhyung Kim[3], Taehyung Kim[3], Yongnyeon Kim[3], Yang Ni[4], Mohsen Imani[4], Ilhong Suh[3], and Yeseong Kim[12*]

***Abstract*— Efficiency and performance are significant challenges in applying Machine Learning (ML) to robotics, especially in energy-constrained real-world scenarios. In this context, Hyperdimensional Computing offers an energy-efficient alternative but has been underexplored in robotics. We introduce ReactHD, an HDC-based framework tailored for perception-action-based learning for sensorimotor controls of robot tasks. ReactHD employs hypervectors to encode sensory inputs and learn the suitable high-dimensional pattern for robot actions. It also integrates two HD-based lightweight symbolic learning techniques: HDC-based supervised learning by demonstration (HDC-IL) and HD-Reinforcement Learning (HDC-RL) to enable precise, reactive robot behaviors in complex environments. Our empirical evaluations show that ReactHD achieves robust and accurate learning outcomes comparable to state-of-the-art deep learning while substantially improving the performance and energy consumption efficiency by 14.2× and 15.3×. To the best of our knowledge, ReactHD is the first HDC-based framework deployed in real-world settings.**

## I. INTRODUCTION

We increasingly seek to harness the capabilities of machine learning(ML) to bolster the autonomy of robotic systems. However, a critical challenge emerges from the current state of ML with its inherent inefficiency in energy consumption and performance on modern GPUs [1].

Brain-inspired Hyperdimensional Computing (HDC) emerges as an innovative computing paradigm to address the concerns surrounding efficiency and lightweight learning [2]. In contrast to conventional numerical computation methodologies that heavily rely on arithmetic operations grounded in Boolean logic, HDC harnesses symbolized information encoded with high-dimensional vectors, taking inspiration from the activities of vast biological neurons. Notably, the machine learning workflows in HDC frequently adopt a binary-centric framework, allowing them to perform highly efficient computations. Prior research has shown substantial performance enhancements, with gains of up to two or three orders of magnitude compared to the performance and energy efficiency of deep learning (DNN) methods [3], [4], [5], [6], [7].

Despite the advances of HDC-based learning algorithms for machine learning tasks [8], [9], [10], a relatively scant body of research has been explored in the context of robotics tasks [11], [12], [13], [14], [15]. They primarily serve as proof-of-concept endeavors to facilitate an automated perception-action mechanism by deploying HDC-based learning methodologies, focusing on emulating the behaviors of robots by mimicking a set of expert inputs, also known as behavior cloning in the form of *imitation learning* (IL). However, the scope of these efforts has often been limited, typically constrained to specific scenarios, such as simulation environments [11], [12], or addressing relatively simplified problems [13], [15].

To initiate on the applications of Hyperdimensional Computing (HDC)-based learning in real-world settings, we must address several technical challenges. Firstly, operating "in the wild" contrasts controlled environments, as the sensor inputs have a high level of complexity. For example, previous work often leverages a *relatively small* number of *discrete* sensor inputs, however, real-world robotic problems necessitate the processing of an extensive array of sensor inputs, e.g., LiDAR sensors, frequently encompassing *continuous* data streams with underlying *inter-feature relationships*. Secondly, a recurring concern with HDC-based solutions in robotics is the potential compromise in accuracy when compared to state-of-the-art ML methodologies represented by DNNs. To enable HDC-based solutions to excel in real robots, it is imperative to mitigate this *accuracy* gap for robust and reliable performance. Lastly, transitioning from supervised learning to a *fully automated* learning solution such as Reinforcement Learning (RL) becomes a crucial requirement, particularly considering the challenges associated with behavior cloning since it often struggles to adapt and generalize in dynamic, real-world scenarios [16].

In this paper, we introduce ReactHD, an advanced HDC-based learning framework tailored for robotic tasks. ReactHD framework first encodes sensory input perceptions to high-dimensional vectors, called *hypervectors*, and learns suitable high-dimensional patterns for potential actions to execute targeted tasks in a notion of perception-action-based learning. In particular, we focus on real-world sensorimotor robot controls that collect LiDAR data and drive it in an eight-directional mode. In tandem with advancements in HDC-based supervised learning for precise IL, called *HDC-IL*, we propose an advanced HDC-based RL solution, called *HDC-RL*, with an emphasis on high sample efficiency. Incorporating RL into the HDC-based framework empowers robots to learn autonomously, adapt to the complexities of dynamic real-world scenarios, and make informed decisions, thereby providing the robustness of real-world robotic applications. Our framework is designed to be capable of addressing diverse problems by simply plugging in task-specific models, such as object tracking, obstacle avoidance, and navigation, on a single platform without requiring additional feature engineering.

[1]Department of Artificial Intelligence, DGIST, Daegu 42988, South Korea
[2]Department of Electrical Engineering and Computer Science, DGIST, Daegu 42988, South Korea
[3]COGA-ROBOTICS, Seoul 04763, South Korea
[4]Department of Computer Science, University of California, Irvine, CA 92697, United States of America
*Corresponding Author: yeseongkim@dgist.ac.kr

To the best of our knowledge, our work is the first pioneering work that deploys an HDC-based solution in the wild unlike the prior work based on simulated environments. In our real-world deployment, we empirically demonstrate highly accurate learning outcomes with HDC while achieving practical reductions in computing energy and resource consumption. We execute all HDC processes on an embedded device, Raspberry Pi 4 based on low-power ARM Cortex-A72, obviating the need for intricate GPGPU support. In our evaluations, we observed that ReactHD achieves comparable learning quality to state-of-the-art deep learning while improving the performance and energy consumption efficiency by 14.2× and 15.3×.

## II. RELATED WORK

HDC is a computing paradigm inspired by the brain designed for efficient learning based on principles from theoretical neuroscience [2], [17]. It has emerged as a novel computing method with several characteristics, such as high computational efficiency and robustness, that as well-suited for robotics applications. There have been a number of attempts to apply HDC to robotics tasks. [12] initially explored the application of robotics tasks in HDC. This was further extended to various tasks, such as sequence-based localization, object recognition tasks [11], and goal-oriented navigation [13]. The work in [15] introduced the concept of transfer learning into HDC-based navigation. While these attempts highlighted the potential of HDC in robotic applications, they were restricted to either simulations or offline dataset-based evaluations. Additionally, those works often utilized a limited number of sensor inputs typically having discrete values.

In contrast to previous work that primarily explored HDC in simulated or highly controlled settings, our paper introduces a pioneering HDC-based learning framework for real-world sensorimotor robotics tasks. By incorporating imitation learning (IL) and reinforcement learning (RL) into the HDC framework, we enable autonomous, adaptive behavior in robots, bridging the gap between computational efficiency and sophisticated real-world applications.

## III. BACKGROUND: MIMICKING HUMAN MEMORY COGNITION WITH HDC

**Hypervector Representation** Inspired by the human brain's ability to associate disparate pieces of *data* and distinguish their *relationships*, HDC hinges on the *hypervector*, a high-dimensional vector with a fixed dimensionality, e.g., $D = 10,000$, i.e., $\mathbf{H} \in \mathbb{R}^D$. To utilize HDC in target learning tasks, we first transform the input data into a set of hypervectors, termed *encoding*. We then proceed to the learning process using operations specifically designed for hypervectors. In this high-dimensional space, randomly generated hypervectors with $D$ components chosen from $\{-1, 1\}$ are nearly orthogonal, implying minimal similarity between them. Each unique item or symbol can thus be captured by a randomly generated hypervector, offering a robust and distributed way to encode information. For instance, if the need arises to represent hypervectors from independent sensors, random hypervectors can be generated for each sensor and employed during the encoding stage.

**Bundling and Binding** HDC employs a set of operations on hypervectors to learn target tasks. These operations emulate human memory functions, like *aggregation* and *linking*, through element-wise operations such as *bundling* and *binding*. Bundling, denoted by $\oplus$, achieves information aggregation by element-wise addition of hypervectors. The resulting hypervector retains the characteristics of its parent vectors, signified by a dot product similarity $\langle \mathbf{H_i}, \mathbf{S} \rangle$ that is significantly greater than zero – approaching $D$ in most cases – where $\mathbf{S} = \mathbf{H}_1 \oplus \cdots \oplus \mathbf{H}_N$. Binding, denoted by $\otimes$, is the element-wise multiplication of hypervectors, leading to a new hypervector that occupies an orthogonal position. Formally, $\langle \mathbf{H_i}, \mathbf{S} \rangle \approx 0$ when $\mathbf{S} = \mathbf{H}_1 \otimes \mathbf{H}_2$, effectively linking two pieces of information with different orthogonal representations. The resulting hypervector can be binarized, i.e., by taking its sign bit, $sign(\mathbf{H})$, to ensure efficiency for the rest of the learning process.

**Permutation** Permutation operations, represented by $\rho^n(\mathbf{H})$, shuffle the components of a hypervector $\mathbf{H}$ by shifting it $n$ bits. The output is a hypervector nearly orthogonal to the original, with $\langle \mathbf{H}, \rho^n(\mathbf{H}) \rangle \approx 0$. The operation's reversibility, $\rho^{-n}(\rho^n(\mathbf{H})) = \mathbf{H}$, makes it useful to represent sequences.

**Reasoning in HDC** HDC's reasoning capabilities hinge on measuring the *similarity* between hypervectors, e.g., the dot product as a metric $\langle \mathbf{H}_1, \mathbf{H}_2 \rangle$. Reasoning involves searching this high-dimensional space to find hypervectors closest to a query. For instance, consider a set of hypervectors that encode sensor information. By bundling the encoded hypervectors according to their corresponding desired actions, we can generate $A$ *action hypervectors* where $A$ is the number of actions. Subsequently, reasoning can be conducted by comparing the similarities between these per-action hypervectors and a query hypervector representing a sensor state.

## IV. HDC LEARNING FOR ROBOTICS

In this paper, we present ReactHD, an HDC-based learning framework specifically engineered for lightweight perception-to-action tasks in the wild. As outlined in Figure 1, the first step is the encoding procedure (❶), which transforms raw input data into hypervectors, considering various characteristics of the real-world sensor inputs. With the encoded hypervector, the objective of our learning procedure is to train action hypervectors (❷) such that they maintain high similarity with the most appropriate hypervectors for each action. To achieve this, we introduce two learning techniques for real-world settings: *IL via supervised learning* (HDC-IL) and *automated learning through RL* (HDC-RL). In HDC-IL, we significantly improve prediction accuracy as compared to the earlier HDC robotics works by iteratively refining the training dataset. i.e., expert behavior logs, employing a strategy akin to state-of-the-art deep neural network algorithms. This also involves considering hard-negative samples, a frequent challenge in many real-world robotic tasks. On the other hand, we also propose an HDC-based RL algorithm, eliminating the need for expert inputs throughout the learning process. Our HDC-based RL algorithm is devised to achieve high sample efficiency, drawing inspiration from the concept of prioritized experience replay [18], but implementing its main concepts thoroughly with HDC operations. Notably, our IL and RL techniques yield HDC models that perform the perception-to-action translation with compatible structures. This architectural consistency offers a compelling advantage of serving HDC models in the deployment (❸) to be readily adapted to various robotics tasks, such as object tracking, obstacle avoidance, and navigation tasks, without structural modification.

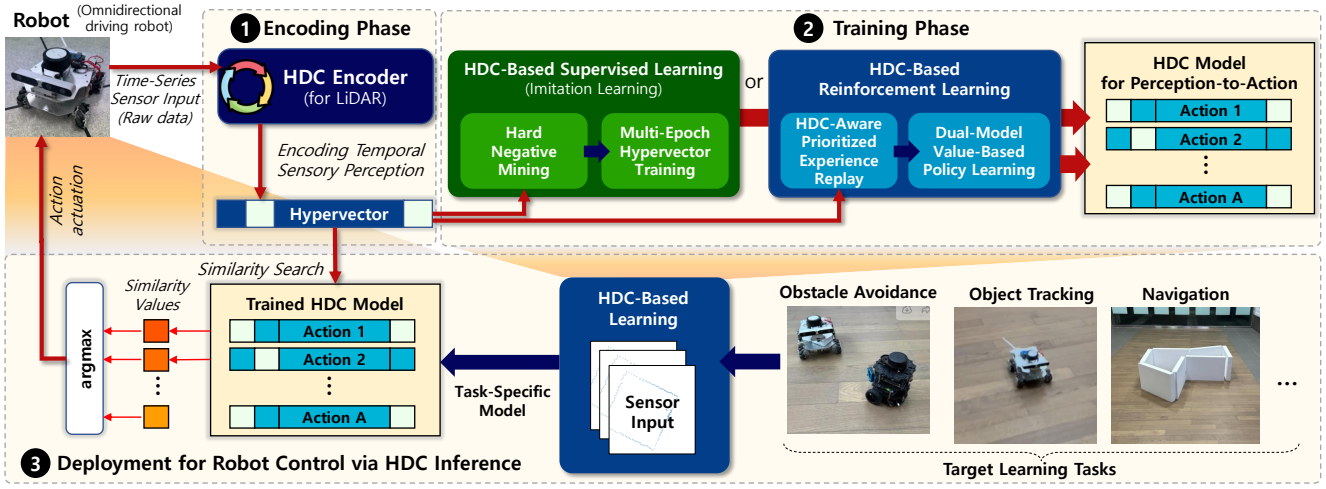



Fig. 1: Overview of ReactHD.

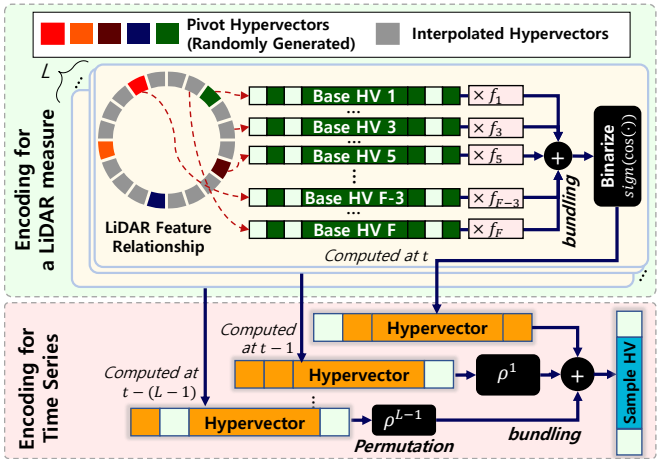



Fig. 2: Proposed Time-Series Encoding for LiDAR.

### A. Data Encoding

The role of the encoding procedure is to project raw features of input data into a hypervector space. The encoding procedure of most of prior HDC works is generally constrained to processing *discrete* categorical values and focuses on *independent* features. For example, in the earlier HDC work for robotics [13], the encoding is represented as $\mathbf{V}_i \otimes \mathbf{B}_i$, where $\mathbf{V}_i$ is a random hypervector associated with a categorical value, and $\mathbf{B}_i$ is another random orthogonal hypervector for a feature position. Unlike the prior works, we develop our encoding strategy with three key considerations for real-world input data: (i) the incorporation of *continuous* values, (ii) the recognition of *inter-feature relationships*, and (iii) the accommodation of temporal ordering. In particular, we focus on how to encode real-world LiDAR inputs.

Figure 2 presents an overview of our encoding scheme, termed *circular encoding*. This mechanism adeptly captures a temporal sequence of LiDAR measurements, each comprising an array of continuous feature values distributed across $F$ directions, i.e., $\vec{F} = \langle f_1, \cdots, f_2 \rangle$. Theoretically underpinned by Random Fourier Features (RFF) [19], a technique validated for its effectiveness in a number of HDC literature [20], [21], [7], [10], [9], our methodology initiates with the generation of a set of $F$ *base hypervectors*, $\mathbb{B} \ni \mathbf{B}_i$. These serve as distinct markers for each feature in the LiDAR data. Assuming we have $F$ LiDAR data points collected at a given moment, our scheme deviates from traditional practices that employ mapping of discrete categorical values to random hypervectors. Instead, we scale each base hypervector with its corresponding feature value. We then employ cosine and sign activation functions to revert the input to a binary space for higher computation efficiency, finally obtaining:

$$\mathbf{H}^t = sign(cos(f_0^t \times \mathbf{B}_0 \oplus \cdots \oplus f_F^t \times \mathbf{B}_F)).$$

In real-world applications, features often exhibit inter-feature relationships, e.g., measurements for the LiDAR data. For example, measurements at proximate degrees tend to be more closely related than those at distant degrees. To account for such relationships explicitly within our encoding framework, we generate base hypervectors that mirror these inherent correlations. Let us introduce a hyperparameter, denoted as $\kappa$, which signifies the number of closely related features that should be encoded with similar base hypervectors. Instead of generating a full set of $F$ base hypervectors, we construct $C = \lceil F/\kappa \rceil$ pivot hypervectors. The other base hypervectors are then synthesized through an interpolation process. The interpolation for a base hypervector, denoted as $\mathbf{B}_{i \cdot \mathcal{K}+\lambda}$ for any $i$, is performed by drawing its initial left $\lfloor (1 - \lambda/\mathcal{K}) \cdot D \rfloor$ elements from $\mathbf{B}_{i \cdot \mathcal{K}}$, while the remaining right elements are then filled in from $\mathbf{B}_{i \cdot (\mathcal{K}+\infty)}$. This method makes the base hypervectors similar to each other among closely situated features. It should be noted that the base hypervectors at each end are similar, thereby reflecting the circular nature of LiDAR measurements.

In the last stage of our encoding scheme, we explicitly incorporate the temporal relationships present in a time series of LiDAR measurements. Leveraging the permutation operation, as elaborated in Section III, we enrich the feature space with temporal context. Specifically, for a time sequence of length $L$, the final encoded hypervector $\mathbf{E}_t$ integrates samples across temporal positions, as captured by the formula: $\mathbf{E}_t = \mathbf{H}^t \oplus \rho^1(\mathbf{H}^{(t-1)}) \oplus \ldots \oplus \rho^{L-1}(\mathbf{H}^{(t-L-1)})$. The permutation operation yields a hypervector orthogonal to its original vector, thus effectively differentiating LiDAR measurements obtained at disparate time intervals.

### B. Imitation Learning Based on Supervised Learning

For imitation learning, ReactHD operates with training dataset curated by experts for a target problem to develop a distinct set of action hypervectors, denoted as $\mathbf{A}_i (\in \mathbb{A})$

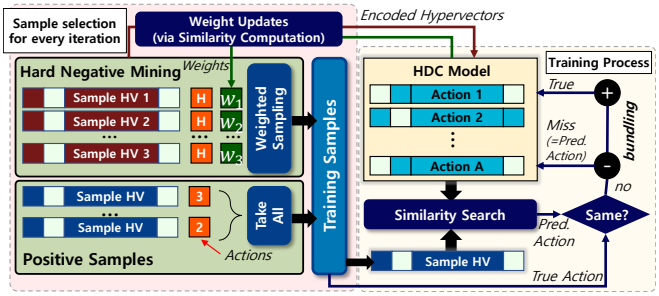



Fig. 3: ReactHD Imitation Learning.

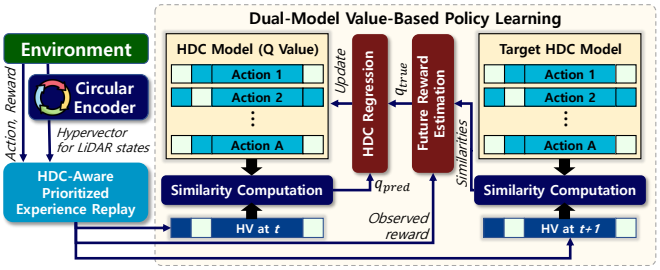



Fig. 4: ReactHD Reinforcement Learning.

where $1 \leq i \leq A$. Figure 3 illustrates the iterative learning procedure tailored for IL, called HDC-IL. This dataset encompasses two key components: (i) feature vectors that are subsequently encoded using our novel circular encoding procedure and (ii) predefined actions governing movement directions. In our specific application, we consider eight directional movements alongside a 'stop' action, yielding a total of $A = 9$ discrete actions.

The traditional HDC-based robot control approaches [13], which generally bundle encoded hypervectors into corresponding action hypervectors $\mathbf{A}_i$ with a single pass until sufficient similarity is achieved, represented as $\langle \mathbf{A}_a, \mathbf{E} \rangle$ where $a$ corresponds to the action for an encoded hypervector $\mathbf{E}$. In contrast, we adopt an iterative learning methodology similar to the DNN techniques, drawing inspiration from a multi-epoch approach, which is called retraining in prior work [9]. During iterative training, the algorithm revisits the training dataset for a predetermined number of iterations. Should a sample be misclassified, both the misclassified and true action hypervectors undergo adaptive adjustments with the bundling operation by:

$$\mathbf{A}_{\text{miss}} = \mathbf{A}_{\text{miss}} - \mathbf{Q} \quad \text{and} \quad \mathbf{A}_{\text{true}} = \mathbf{A}_{\text{true}} + \mathbf{Q}.$$

Another key piece of our learning algorithm involves the methodical management of hard negative samples to refine action hypervectors. In real-world scenarios, expert actions are not uniformly distributed. Consider, for instance, the task of obstacle avoidance. It is inclined to predominantly select the 'stop' action due to the frequent absence of close objects. As a result, in our dataset representing the inputs of the human expert, the frequency of 'stop' actions is on average 7.33 times greater than that of any other action. In this context, hard-negative samples are particularly challenging instances where the model incorrectly predicts 'non-stop' actions in situations that necessitate a 'stop'.

To address this issue, we introduce a hard-negative mining technique tailored with HDC operations, parameterized by the hypervector $\alpha$, as outlined in Algorithm 1. Before each training iteration, we utilize the current state of the HDC action model $\mathbb{A}$ to compute high-dimensional similarities for all encoded samples, $\mathbf{E}_i^H \in \mathbb{E}^H$, labeled as the negative class $H$, e.g., 'stop'. The softmax function is applied to the similarity vector for each sample to normalize, which is then subtracted from a one-hot vector identifying the target negative action, i.e., $\vec{O} = [\cdots o_i \cdots]$ where $o_i = 1$ if $i = H$ otherwise zero. The resulting mean serves as a weight for each sample, indicating its difficulty to classify correctly. Through weighted sampling with a list of weights $\mathcal{W}$, we selectively incorporate these hard-negative samples in the training iteration, effectively balancing the training set. The number of hard-negative samples included is calculated as $\alpha \times \widetilde{N}$, where $\widetilde{N}$ represents the average count of samples for each non-negative action. As a result, the training algorithm proceeds to focus more on the hard-negative samples throughout the iterative procedure, creating the robots model for real-world scenarios.

**Algorithm 1** HDC-based Hard-Negative Mining

**Input:** $\mathbf{A}_i \in \mathbb{A}$, $\mathbb{E}^H$, $\alpha$, $\widetilde{N}$
**Output:** Hard-negative samples for training iteration
**Initialize:** Weight vector $\mathcal{W} \leftarrow []$
**for** each hypervector $\mathbf{E}_j^H \in \mathbb{E}^H$ for action $H$ **do**
    $\vec{V} = \vec{O} - \text{softmax}([\cdots \langle \mathbf{E}_j^H, \mathbf{A}_i \rangle \cdots])$
    $\mathcal{W} += mean(\vec{V})$
**end for**
Compute the weighted sampling rate as $\alpha \times \widetilde{N}$
Sample $\alpha \times \widetilde{N}$ hard-negative samples from $\mathcal{W}$

### C. Reinforcement Learning

IL is known to be inherently limited by the scope of the training data and cannot adapt to unseen situations. To address these limitations, ReactHD incorporates RL to enable the agent to interact with the environment and learn optimal behavior through trial and error, called HDC-RL. As shown in Figure 4, operating in a typical episodic environment providing the states and rewards and actuating with given actions, our HDC-based RL approach draws parallels from the recent development of the HDC-based RL algorithm [22] with crucial modifications for the sample efficiency for real-world scenarios. Specifically, we maintain two copies of the action models: one for Q-value estimation later (also later used for deployment) and another serving as a periodically updated target model to stabilize learning called the target HDC model, say $\mathbf{A}_i \in \mathbb{A}$ and $\mathbf{A'}_i \in \mathbb{A}'$, respectively. Furthermore, we utilize a replay buffer to store experiences, facilitating efficient training and addressing the non-stationary nature of the RL problem. The replay buffer stores tuples of experiences $(\mathbf{E}_t, a_t, r_t, \mathbf{E}_{t+1})$ where $\mathbf{E}_t$ and $\mathbf{E}_{t+1}$ represent the encoded hypervectors for the current and next states, $a_t$ is the action taken, and $r_t$ is the received reward. These tuples are sampled in mini-batches to update the action hypervectors during training. In ReactHD, we incorporate the dual-model and replay-buffer mechanisms used in the original DQN [23] into the HDC learning.

The proposed RL algorithm learns a value-based policy to maximize the resultant Q-values. Our goal is to train the action hypervectors $\boldsymbol{A}_i$ so that the predicted Q-value, denoted as $q_{\text{pred}}$, is obtained through the dot product similarity between $\boldsymbol{A}_i$ and the encoded hypervector for the current state $\boldsymbol{E}_t$: $q_{\text{pred}} = \langle \mathbf{A}_i \cdot \mathbf{E}_t \rangle$. To this end, the two sets of action hypervectors $\mathbb{A}$ and $\mathbb{A}'$ are initialized as zero vectors. During each training iteration, a mini-batch $\mathcal{M}$ is sampled from $\mathcal{B}$ and is used to update the action hypervectors in $\mathbb{A}$

according to the error between the predicted Q-value, $q_{\text{pred}}$, and the ground truth Q-value, $q_{\text{true}}$. The update equation for the action hypervector corresponding to a specific action $a_t$ with a learning rate $\beta$ is formalized as:

$$\boldsymbol{A}_{a_t} = \boldsymbol{A}_{a_t} \oplus \beta(q_{\text{true}} - q_{\text{pred}}) \times \boldsymbol{E}_t.$$

To calculate $q_{\text{true}}$, a target model $\mathbb{A}'$ is employed. Specifically, $q_{\text{true}}$ is computed with a discount rate $\gamma$ as:

$$q_{\text{true}} = r_t + \gamma \max_a \langle \mathbf{A'}_a, \mathbf{E}_{t+1} \rangle.$$

The target model $\mathbb{A}'$ is updated at predefined intervals, ensuring a more stable convergence of the Q-values. To balance the exploration and exploitation, an $\epsilon$-decay policy is instituted. Upon action selection, the RL agent interacts with its environment, consequently acquiring a new state and a corresponding reward. These sequences of state-action-reward-state transitions are organized into episodes, serving as training data to refine the RL action model iteratively.

In our evaluation of the initial RL algorithm, we found that learning in LiDAR-based robotic environments often progressed slowly, primarily due to the agent's regular interaction with less informative samples. This issue arises from the complexity of LiDAR data, which contains hundreds of features, thus significantly expanding the problem space, in contrast to the environments with fewer than 10 features typically explored in existing HDC-based RL research [22]. To address this issue, we integrate a prioritized experience replay mechanism by translating its main conception with HDC operations. To this end, our algorithm utilizes a 'surprise' factor for each sample, which is quantified during the action selection as the difference in dot-product similarity between $q_{\text{pred}}$ and $q_{\text{true}}$. This factor is stored in the replay buffer alongside the respective experience tuples. For training, it takes a mini-batch $\mathcal{M}$ using weighted sampling based on this 'surprise' factor, thereby skewing the distribution of sampled experiences toward more informative states. The original concept of importance-sampling weights, used in conventional prioritized experience replay [18], is incorporated by multiplying it with $\beta$. During the post-training, the factors for the used samples are updated in the replay buffer, ensuring an adaptive learning process with high sample efficiency.

## V. EXPERIMENTAL RESULTS

We employ a Wheeltec Mecanum Omnidirectional driving robot with a Raspberry Pi 4, which leverages a low-power ARM Cortex-A72 for computational tasks. The robot is equipped with an RPLidar A1 Lidar sensor, operating with a scan resolution of 0.5°, equating to 720 data points per scan and a scan frequency ranging between 8 to 9 Hz. It thus obtains $F = 720$ features for Lidar reading, and we pass them to the encoding process that considers $L = 7$ successive time-series measures. For the robot's mobility, we define a set of nine actions that allow control in eight directions along with a 'stop' action. The HDC-IL and HDC-RL algorithms and other comparative methods are implemented using PyTorch 1.12 running on CPU.

In addition to the physical experiments, we develop an in-house simulation environment designed to emulate the dynamics of the omnidirectional driving robot within various obstacle-rich settings and synthesize LiDAR data. By leveraging this simulated environment, we can rigorously pre-train our HDC-based RL models without the inefficiency and potential safety risks associated with direct RL training on the actual robotic hardware. We then deployed the trained HDC models onto the physical robot for evaluation.

**Evaluated Tasks:** We evaluated five tasks designed to address different challenges in robotics. We evaluate the performance of both our HDC-based IL and RL approach. For IL, we use the data collected for the actual robot driving:

• **Obstacle Avoidance** (AVO): This task requires the robot to maintain a predefined safe distance (50 cm) from a moving obstacle. The IL model is trained on data collected for 6 minutes long, while the RL model is structured to provide a single -1 reward for proximity violations.

• **Object Tracking** (TRA): This task has a similar setup to the obstacle avoidance task; the goal is to track the closest person within a certain distance threshold.

• **Navigation 1 & 2**[1] (NAV1 & NAV2): In NAV1, the robot is tasked with making a circle around a set of centrally-placed boxes. NAV2 features a map scattered with various objects. The robot aims to move from a starting point to an endpoint through a more challenging environment. We utilized the training data for eight successful trials for IL. For RL, we give positive rewards when passing each corner of major objects to help navigation, while a negative reward is given when hitting a wall.

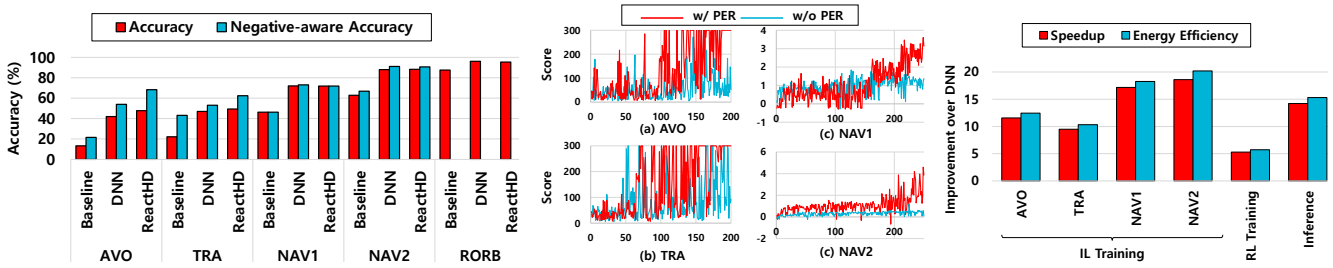
In addition to four tasks with real-world robots, we include another task mainly for comparative assessment, called RORB. This task is taken from prior literature that proposed a concept for HDC-based virtual robot controls [13]. It employs a simplified 10 x 10 2-D grid environment with 15 obstacles randomly located. A virtual robot moves one grid cell at a time in one of the four cardinal directions. It relies on six types of sensor inputs: four for detecting obstacles in each direction and two for denoting the x and y directions toward the goal. An additional feature is also included to consider the robot's last action, i.e., having seven input features $F = 7$ with four direction movement.

### A. Learning Quality Evaluation

**Imitation Learning** We first assess the learning efficacy of HDC-IL with comparisons with a traditional HDC algorithm and a state-of-the-art Deep Neural Network (DNN) consisting of three fully-connected layers, each containing 512 neurons. The baseline HDC algorithm represents earlier HDC-based approaches for robotics tasks [13], relying on simplistic binding-to-bundling operations and randomly generated hypervectors for encoding [8] under the assumption that all sensor inputs are independent, and (ii) employing single-pass learning for its training. To address the challenges of highly imbalanced action scenarios in robotics, we utilize a custom metric named 'negative-aware accuracy'. This metric regards predictions as correct for non-stop action samples if they match the true label or are predicted as 'stop'. Conversely, for samples labeled as 'stop', accuracy requires precise prediction as 'stop'. This metric is employed for safe robot decision-making by prioritizing the recognition of inaction when uncertain.

As illustrated in Figure 5a, our proposed IL algorithm demonstrates superior performance over the baseline HD algorithm. Our approach yields an average accuracy that is 24.3% higher across four real-world LiDAR-based tasks. Notably, in the context of the four LiDAR-based tasks, our

[1] Visual representations for the navigation tasks are shown in Section V-C.




(a) Learning Quality Evaluation for IL (b) RL Learning Quality Comparison (c) Efficiency Evaluation

Fig. 5: Quantitative Evaluation of ReactHD

approach effectively navigates the challenges presented by the extensive feature space ($F = 720$), a scenario where the baseline algorithm struggles. The robustness of our method to hard negatives ensures it avoids the pitfalls of defaulting to ineffective negative stop actions, significantly improving the negative-aware accuracy. As compared to the deep learning model, our algorithm maintains a comparable learning quality. The advantages of HDC-based strategies, such as their computationally efficient nature, as compared to the DNN will be further elaborated in Section V-B.

**Reinforcement Learning** Next, we evaluate our HDC-RL algorithm.[2] Figure 5b compares the scores obtained over multiple episodes. Here, the score is defined as the number of successful steps for AVO and TRA, and as the mean reward accumulated over steps per episode for NAV1 and NAV2. Additionally, we evaluate our algorithm without the use of prioritized experience replay. Although this simplified version can accomplish some tasks, it generally requires longer run times and sometimes fails to learn within an acceptable number of episodes. This limitation arises from its inability to adequately prioritize samples in the replay buffer, a challenge that our proposed algorithm successfully overcomes from existing HDC-based RL methods [22].

### B. Efficiency Evaluation

In this section, we discuss the system efficiency of learning-based robotics tasks, particularly when running our proposed HDC-based algorithm and the DNN model on a Raspberry Pi (RPi) device, which has computational and power limitations often encountered in robotic environments. Figure 5c presents the improvement of performance and energy efficiency as compared to DNN. For instance, in training imitation learning (IL), it achieves an average speedup of $14.2\times$ and an energy efficiency improvement of $15.3\times$. To train a minibatch in RL, the HDC-RL archives a speedup by $5.3\times$. For inference, we assess the time it takes to process a single LiDAR measure in real-time. Notably, the inference runtimes for both IL and RL models are identical, regardless of the target tasks. The results show that the ReactHD framework also significantly enhances inference efficiency, being $14.2\times$ faster than the DNN.

### C. Ablation Study

**HDC-IL** We next evaluate the contributing factors to the learning quality improvements in our proposed imitation learning approach. To this end, we experiment with two altered versions of the algorithm: one without using the circular encoding and another without using the hard negative mining. As shown in Figure 6a, both elements play a crucial role in enhancing learning quality. For instance, when circular encoding is excluded, there is a significant decrease in accuracy by 12.5% across all four tasks. The hard negative mining is also an important component, particularly for specific robotics tasks[3], where it yields improvements of 5.8% for TRA.

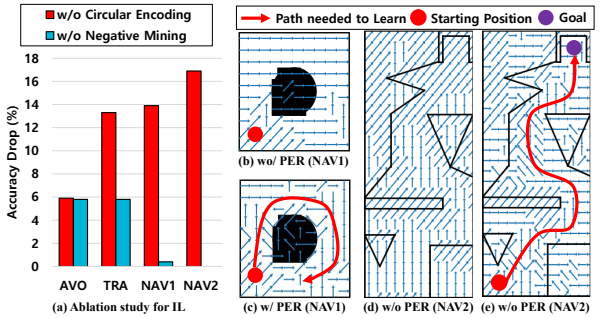



Fig. 6: Ablation Study. The dominant actions are evaluated after 30 epochs for NAV1 and 150 epochs for NAV2.

**HDC-RL** We here assess the impact of prioritized experience replay by comparing the trajectories in navigation tasks. Figure 6b-e presents the learned actions having the highest Q value at each grid point on the navigation map after learning across different numbers of episodes. The results show that our original approach, as presented with the detailed decision-making for different grids, achieves faster convergence on the navigation tasks. In contrast, the approach without using prioritized experience replay (PER) learns a relatively small portion of the map, eventually failing to learn NAV2 after 1000 epochs. We believe that this enhanced learning speed significantly increases the practicality of our method for real-world applications, as it enables the HDC algorithm to prioritize challenging and informative experiences, thereby improving the overall quality and efficiency of learning.

## VI. CONCLUSIONS

We present ReactHD, an HDC-based learning framework devised specifically for real-world robotic tasks. Based on a new HDC encoding method for real-world LiDAR sensors, we proposed HDC-based IL and RL for accurate learning in the wild. We evaluated ReactHD's capability along with real-world deployment and show that it achieves high learning quality while significantly reducing computational and energy requirements, e.g., improving the inference performance by $14.2\times$.

[2]We also verify the trained models with the real-world deployment. Please refer to the accompanying video.

[3]Note that the navigation tasks do not experience significant effects since the expert inputs consist of consistent movements.

## REFERENCES


[1] V. Sze, Y.-H. Chen, J. Emer, A. Suleiman, and Z. Zhang, "Hardware for machine learning: Challenges and opportunities," in *2017 IEEE Custom Integrated Circuits Conference (CICC)*, pp. 1–8, 2017.

[2] P. Kanerva, "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors," *Cognitive Computation*, vol. 1, no. 2, p. 139–159, 2009.

[3] S. Gupta, M. Imani, and T. Rosing, "Felix: Fast and energy-efficient logic in memory," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–7, IEEE, 2018.

[4] H. Li, T. Wu, A. Rahimi, K.-S. Li, M. Rusch, C. Lin, J.-L. Hsu, M. Sabry, S. Eryilmaz, J. Sohn, W. Chiu, M.-C. Chen, T.-T. Wu, J. Shieh, W. Yeh, J. Rabaey, S. Mitra, and H. Wong, "Hyperdimensional computing with 3d vrram in-memory kernels: Device-architecture co-design for energy-efficient, error-resilient language recognition," *2016 IEEE International Electron Devices Meeting (IEDM)*, pp. 16.1.1–16.1.4, 2016.

[5] S. Gupta, J. Morris, M. Imani, R. Ramkumar, J. Yu, A. Tiwari, B. Aksanli, and T. Š. Rosing, "Thrifty: training with hyperdimensional computing across flash hierarchy," in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–9, IEEE, 2020.

[6] 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 Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 3, pp. 439–452, 2019.

[7] M. Imani, S. Salamat, S. Gupta, J. Huang, and T. Rosing, "Fach: Fpga-based acceleration of hyperdimensional computing by reducing computational complexity," in *Proceedings of the 24th Asia and South Pacific Design Automation Conference*, pp. 493–498, 2019.

[8] 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)*, pp. 1–8, 2017.

[9] Y. Kim, J. Kim, and M. Imani, "Cascadehd: Efficient many-class learning framework using hyperdimensional computing," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 775–780, 2021.

[10] Z. Zou, Y. Kim, M. H. Najafi$\psi$, and M. Imani, "Manihd: Efficient hyper-dimensional learning using manifold trainable encoder," *DATE, IEEE*, 2021.

[11] P. Neubert, S. Schubert, and P. Protzel, "An introduction to hyperdimensional computing for robotics," *KI - Künstliche Intelligenz*, vol. 33, no. 4, p. 319–330, 2019.

[12] P. Neubert, S. Schubert, and P. Protzel, "Learning vector symbolic architectures for reactive robot behaviours," 2017.

[13] A. Menon, A. Natarajan, L. I. G. Olascoaga, Y. Kim, B. Benedict, and J. M. Rabaey, "On the role of hyperdimensional computing for behavioral prioritization in reactive robot navigation tasks," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 7335–7341, 2022.

[14] A. Mitrokhin, P. Sutor, C. Fermüller, and Y. Aloimonos, "Learning sensorimotor control with neuromorphic sensors: Toward hyperdimensional active perception," *Science Robotics*, vol. 4, no. 30, p. eaaw6736, 2019.

[15] N. McDonald, R. Davis, L. Loomis, and J. Kopra, "Aspects of hyperdimensional computing for robotics: transfer learning, cloning, extraneous sensors, and network topology," in *Disruptive Technologies in Information Sciences V*, vol. 11751, pp. 21–33, SPIE, 2021.

[16] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, J. Peters, *et al.*, "An algorithmic perspective on imitation learning," *Foundations and Trends® in Robotics*, vol. 7, no. 1-2, pp. 1–179, 2018.

[17] P. Kanerva, *Sparse distributed memory*. MIT press, 1988.

[18] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[19] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, (Red Hook, NY, USA), p. 1177–1184, Curran Associates Inc., 2007.

[20] M. Imani, S. Pampana, S. Gupta, M. Zhou, Y. Kim, and T. Rosing, "Dual: Acceleration of clustering algorithms using digital-based processing in-memory," in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 356–371, 2020.

[21] B. Khaleghi, H. Xu, J. Morris, and T. R. Rosing, "tiny-hd: Ultra-efficient hyperdimensional computing engine for iot applications," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 408–413, 2021.

[22] Y. Ni, D. Abraham, M. Issa, Y. Kim, P. Mercati, and M. Imani, "Efficient off-policy reinforcement learning via brain-inspired computing," *arXiv preprint arXiv:2205.06978*, 2022.

[23] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.