Classifying Functional Brain Graphs using Graph Hypervector Representation

Lulu Ge
Dept. of ECE
University of Minnesota
Minneapolis, MN, USA
ge000567@umn.edu

Ali Payani
Cisco Research
San Jose, CA, USA
apayani@cisco.com

Hugo Latapie
Cisco Research
San Jose, CA, USA
hlatapie@cisco.com

Keshab K. Parhi
Dept. of ECE
University of Minnesota
Minneapolis, MN, USA
parhi@umn.edu

***Abstract*—Hyperdimensional computing (HDC) has drawn significant attention due to its comparable performance with traditional machine learning techniques. HDC classifiers achieve high parallelism, consume less power, and are well-suited for edge applications. Encoding approaches such as record-based encoding and N-gram-based encoding have been used to generate features from input signals and images. These features are mapped to hypervectors and are input to HDC classifiers. This paper considers the group-based classification of graphs constructed from time series. The graph is encoded to a hypervector and the graph hypervectors are used to train the HDC classifier. This paper applies HDC to brain graph classification using fMRI data. Both the record-based encoding and GrapHD encoding are explored. Experimental results show that *1)* For HDC encoding approaches, GrapHD encoding can achieve comparable classification performance and require significantly less memory storage compared to record-based encoding. *2)* The utilization of sparsity can achieve higher performance as compared to fully connected brain graphs. Both threshold strategy and the minimum redundancy maximum relevance (mRMR) algorithm are employed to generate sub-graphs, where mRMR achieves higher performance for three binary classification problems: emotion *vs.* gambling, emotion *vs.* no-task, and gambling *vs.* no-task. The corresponding AUCs are 0.87, 0.88, and 0.88, respectively.**

***Index Terms*—Hyperdimensional computing (HDC), classification, record-based encoding, graph hypervector, brain state classification.**

I. Introduction

Hyperdimensional computing (HDC), proposed in 1988 [1], also known as vector symbolic architecture (VSA) [2], continues to attract considerable interest. Compared to traditional machine learning and deep learning, HDC has its unique data representation, data transformation, and data interpretation. Instead of computing scalar numbers, HDC manipulates its ultra-long vectors to represent the patterns of neural activities. Those vectors are referred to as *hypervectors* with their dimensionality in the thousands, e.g., d = 10,000, where d denotes the dimensionality. Such a high-dimensional vector can be used to represent all possible states of a human brain, which on average contains about 100 billion neurons and 1,000 trillion synapses. Data transformation for HDC is typically realized by three types of bit-level arithmetic operations, namely multiply-add-permute (MAP) [3]. Similarity measurement is performed to relate or distinguish different patterns. In this sense, HDC is a brain-inspired computing paradigm that mimics how the human brain works [3]. Although still in its infancy, HDC continues to draw significant attention from both academia and industry due to its comparable performance with traditional machine learning techniques. HDC classifiers achieve high noise immunity, massive parallelism, high energy efficiency, fast learning/inference speed, and one-/few-shot learning ability. So far, the potential of HDC has been demonstrated for applications in cognitive tasks, e.g., language recognition [4], speech recognition [5], image classification [6], DNA sequencing [7] and bio-signal processing [8].

The *state* of the human brain network dynamically changes from task to task or from resting state (no-task) to the task. Here the state refers to a specific pattern in brain connectivity [9]. Researchers have great interest in understanding brain connectivity patterns through the lens of network theory. The corresponding hidden assumptions are *1)* The whole brain network can be denoted as a functional brain graph $G(V, E)$, where V and E, respectively, represent the vertices/nodes and the edges. *2)* Although the brain states change dynamically when tasks or no-tasks are conducted, each state has its specific pattern, namely the brain graph. *3)* Network metric is group differentiating and biologically meaningful. Built on these three assumptions, [9] employs novel features—sub-graph entropies—to classify three brain graph classification problems (emotion *vs.* gambling, emotion *vs.* no-task, and gambling *vs.* no-task) using the fMRI data, which are publicly available in the Human Connectome Project (HCP) study [10].

Most HDC applications deploy two encoding approaches: record-based encoding and N-gram-based encoding [3]. Both of these have been applied to efficiently represent the data structure like "sequence" [3], [8], [11]. A novel encoding approach, called GrapHD, has been recently proposed to encode the graph structure [12] as a graph hypervector. In [12], storage and retrieval of graphs using HDC and graph matching are considered.

Whether two classes of graphs can be discriminated based on their GrapHD representation has not been investigated. This paper addresses the group-based classification of graphs using

This paper was supported in parts by NSF grant CCF-1814759, by CISCO Systems, and by the Chinese Scholarship Council (CSC).

GrapHD. In particular, we classify functional brain graphs associated with different tasks. We compare the performance of the GrapHD encoding-based classification with the classical record-based encoding using HDC and traditional support vector machine (SVM). Based on the experimental results, these two HDC encoding approaches require further/additional steps to improve the classification performance since their corresponding performance is lower than traditional linear SVM. Using sub-graphs can improve the performance of HDC. Moreover, GrapHD is preferred because record-based encoding requires $\sim 41\times$ larger memory storage.

The main contributions of this paper are three-fold. First, using correlation coefficients as edge weights, we show that graph hypervectors lead to the same accuracy as with record-based encoding. Second, GrapHD was used for storage and retrieval in prior work [12]. In this paper, we extend the applicability of GrapHD to for classification. Third, as compared to fully connected brain graphs, sub-graphs can improve the performance of HDC classifiers.

The remainder of this paper is organized as follows. Sec. II introduces the HCP dataset and presents an overview of HDC-based classification. Sec. III presents two encoding approaches using HDC: record-based and GrapHD. To be more specific, the record-based encoding flattens the matrix to generate a one-dimensional vector of edge weights, whereas the GrapHD is used to encode the graph structure. The corresponding experimental results are discussed in Sec. IV. To further improve the performance of GraHD, two strategies for generating the sub-graphs are employed. Both the threshold strategy and the minimum redundancy maximum relevance (mRMR) algorithm are investigated. Finally, Sec. V summarizes the entire paper.

II. Preliminaries

This section presents the basics of HCP dataset. This is followed by an overview of HDC-based classification. A flowchart of the approaches using HDC for this paper is also included.

A. HCP Dataset and Preprocessing

In this work, we use the publicly available dataset from the HCP [9], where seven different task-fMRI data are collected from 475 healthy subjects. Similar to [9], two chosen tasks are investigated in this paper: gambling and emotion. One thing that should be emphasized is that no-task fMRI data, indicating the resting state, are also used as a baseline. For each state (both task and no-task), the corresponding fMRI data are acquired on a scanner for two runs, from right to left (RL) and from left to right (LR). The reader is referred to [9] for a more detailed description of the data.

Pre-processing should be conducted to model the brain graph from fMRI. Using the scanner, two fMRI time series are acquired for two runs. Then a matrix of $R \times T$ is obtained after averaging the two runs of the time series from the predefined anatomical regions from fMRI, where R is the number of regions and T is the number of time points. In the brain graph, each node corresponds to a region of interest (RoI), whereas each edge weight is the absolute value of the Pearson correlation coefficient between the time series of the related two nodes. Due to this reason, the edge weight $w \in [0, 1]$.

Similar to [9], this paper essentially addresses the brain graph classification from fMRI data at a group level, where only 85 RoIs are considered. However, unlike [9] where entropy measures are used as features, this paper considers a graph where the edge weight corresponds to the absolute value of the correlation coefficient.

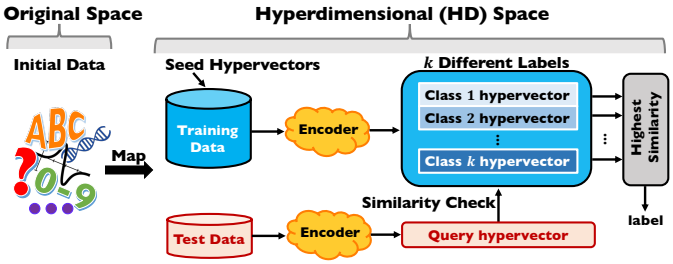


Fig. 1. Classification overview for HDC [3].

B. Classification Overview of Hyperdimensional Computing

There are three different types of *seed* or *atomic* hypervectors: *random-*, *level-* and *circular*-hypervectors [13]. Each type of the seed hypervector possesses its unique correlation profile. These properties include: *1)* Random-hypervectors are quasi-orthogonal to each other and are typically used to represent the independently categorical/symbolic data, e.g., 26 alphabets; *2)* Level-hypervectors are usually linearly correlated and are employed to encode the sub-intervals of a certain range, e.g., quantization of the given range [0,1] into several levels; *3)* Circular-hypervectors are a variant of level-hypervectors and are mostly used to handle the cyclic intervals, such as Jan 1st - Dec 31st [13], [14]. In this paper, only the random- and level-hypervectors are considered. Interested readers are referred to [3], [13] for more details.

As shown in Fig. 1, starting from seed hypervectors, HDC employs a certain encoding approach that maps the original input data into HD space. Such an encoding approach involves mathematical operations—addition (+), multiplication (*) and permutation (ρ). After the seed hypervectors are manipulated, meaningful compound hypervectors are generated at the output.

For any HDC classification problem, during the training phase, *class* hypervectors are generated using a certain encoding approach. During the inference phase, a *query* hypervector is generated based on an unknown test data. The predicted label is determined by the highest similarity between the query and pre-trained class hypervectors. In terms of the similarity measurement, Hamming distance is adopted for binary HDC, whereas cosine similarity is employed for non-binary HDC. The experimental results throughout this paper are presented using cosine similarity (δ) as calculated in (1), where d is the dimensionality of the two hypervectors $\mathbf{A}$ and $\mathbf{B}$, and $\cdot$ denotes the dot product.

$$\delta(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{d}. \tag{1}$$

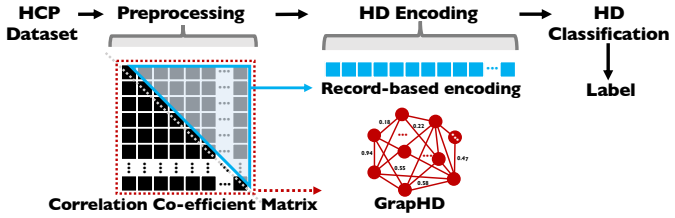


Fig. 2. Flowchart of the two HDC approaches for the brain state classification.

In this paper, the dimensionality of the hypervectors is $d = 10,000$. The seed hypervectors are composed of bipolar values ($\in \{-1, 1\}^d$).

C. Flowchart of The Approaches

Fig. 2 illustrates how the two HDC encoding approaches are applied for the brain graph classification using the fMRI data in the HCP study. Following the setup of [9], pre-processing should be conducted over the time-series to generate the functional connectivity, namely the 85×85 correlation-coefficient matrix, where 85 is the number of RoI (refer to [9]). Using this matrix as the input features, HD encoding is performed by constructing the data structure. Record-based encoding is applied when the "sequence" data structure is extracted. GrapHD is adopted once the graph structure is built. HDC classification is carried out after the information is efficiently encoded.

III. Methodology and Workflow

In this section, we illustrate two HDC approaches to encode brain graphs: record-based and GrapHD.

A. Record-based Encoding

Record-based encoding is typically used to represent the information whose data structure is in a "one-to-one mapping" form, e.g., a key-value pair [15]. For this specific problem, the correlation coefficient matrix is the input feature. Since this matrix is symmetric along the diagonal, i.e., coefficient value $c_{ij} = c_{ji}$, only the upper-/lower-triangular information (excluding the diagonal) is required. Then, the effective feature becomes a one-dimensional vector after the triangular data are flattened. At the very beginning, seed hypervectors, $\{\mathbf{L_1}, \cdots, \mathbf{L_q}\}$ and $\{\mathbf{ID}_1, \cdots, \mathbf{ID}_{\binom{m}{2}}\}$, should be pre-generated to represent both the coefficient values and position indices, respectively. Here q represents the quantization level and m is the number of nodes. The former hypervectors can be either generated by level-hypervectors (see [16]) or the *edge weight* hypervectors $\mathbf{E}_w$ (Sec. III-B), whereas the latter ones are random-hypervectors. Then record-based encoding is applied to this data structure as follows: *1)* Each position value is encoded by $\bar{\mathbf{V}}_k$, which corresponds to the coefficient values c_{ij} with the relationship described by $\phi(c_{ij}) = \bar{\mathbf{V}}_k$. As shown in (2a), the value hypervector $\bar{\mathbf{V}}_k$ can be picked from the seed hypervectors $\{\mathbf{L_1}, \cdots, \mathbf{L_q}\}$ (or $\{\mathbf{E}_w\}$) based on its coefficitent value. *2)* The compound hypervector $\mathbf{S}_i$ for this vector can be calculated by (2b). *3)* The class hypervector $\mathbf{hv_{class}}$ is formed by adding its constituent hypervectors $\mathbf{S}_i$ as described in (2c).

$$\bar{\mathbf{V}}_k \in \{\mathbf{L_1}, \cdots, \mathbf{L_q}\}, \quad (\text{ or } \bar{\mathbf{V}}_k = \mathbf{E}_w), \tag{2a}$$

$$\mathbf{S}_i = \bar{\mathbf{V}}_1 * \mathrm{ID}_1 + \cdots + \bar{\mathbf{V}}_k * \mathrm{ID}_k + \cdots + \bar{\mathbf{V}}_{\binom{m}{2}} * \mathrm{ID}_{\binom{m}{2}}, \tag{2b}$$

$$\text{where } 1 \leq k \leq \binom{m}{2} \text{ if the given matrix is } m \times m,$$

$$\mathbf{hv_{class}} = \textstyle\sum_{i=1}^{N} \mathbf{S}_i, \text{ where } N \text{ is the number of subjects.} \tag{2c}$$

B. GrapHD Encoding

GrapHD is proposed to represent an arbitrary graph which is composed of nodes and edges [12]. Such an encoding approach can represent: *1)* unweighted undirected, *2)* unweighted directed, *3)* weighted undirected, and *4)* weighted directed graphs. For this HCP dataset, the functional connectivity for each subject—essentially an 85×85 correlation-coefficient matrix—is extracted from the corresponding fMRI data. Based on this matrix, a weighted undirected graph can be constructed.

Fig. 3 illustrates how to encode an arbitrary weighted undirected graph with m nodes using GrapHD. *1)* Generate a total of $(m + 1)$ seed hypervectors, where node hypervectors ($\{\mathbf{N}_1, \mathbf{N}_2, \cdots, \mathbf{N}_m\}$) correspond to the m nodes and 1 edge weight hypervector ($\mathbf{E}_{\text{"1"}}$) represents the weight value "1". For the other weight values w, the corresponding hypervectors $\mathbf{E}_w$ are generated by flipping the $(\frac{1+w}{2}d)^{th} \sim d^{th}$ rightmost bits of the hypervector $\mathbf{E}_{\text{"1"}}$. This flipping operation ensures that the cosine similarity between w and the 1 vector, $\mathbf{E}_{\text{"1"}}$, is w (see (4)). *2)* The node memory $\mathbf{M}_k$ reflects the edge information regarding the node k as shown in (3b). *3)* The whole graph is encoded as described in (3c). Note that the query hypervectors can be generated from these three steps (3a, 3b,3c), whereas class hypervectors are obtained by (3).

$$\mathbf{N}_k \in \{\mathbf{N}_1, \mathbf{N}_2, \cdots, \mathbf{N}_m\}, \text{ weight value "1": } \mathbf{E}_{\text{"1"}} \tag{3a}$$

$$\mathbf{M}_k = \textstyle\sum_k \mathbf{E}_{w_{kj}} * \mathbf{N}_j, \text{ where } j \in \{\text{nodes adjacent to node } k\}, \tag{3b}$$

$$\mathbf{G}_i = \frac{1}{2} \sum_k \mathbf{M}_k * \mathbf{N}_k, \text{ where } 1 \leq k \leq m, \tag{3c}$$

$$\mathbf{hv_{class}} = \textstyle\sum_{i=1}^{N} \mathbf{G}_i, \text{ where } N \text{ is the number of subjects.} \tag{3d}$$

$$\delta(\mathbf{E}_w, \mathbf{E}_{\text{"1"}}) = \frac{\mathbf{E}_w \cdot \mathbf{E}_{\text{"1"}}}{d} = \frac{\frac{1+w}{2}d - \overbrace{\frac{1-w}{2}d}^{\text{flip these bits}}}{d} = w \tag{4}$$

C. Training and Test Workflow

We conduct leave-one-subject-out cross-validation for three binary classification problems, i.e., gambling *vs.* emotion, gambling *vs.* no-task, and emotion *vs.* no-task. For N subjects, there are N folds. Within each fold, the HD model learns from (N–1) subjects (training data) and then predicts the remaining 1 subject (test data).

IV. Experimental Results

A. Evaluation Metrics

For this classification problem, following a similar evaluation setting in [9], four metrics are measured for the performance evaluation: accuracy, sensitivity, specificity, and area

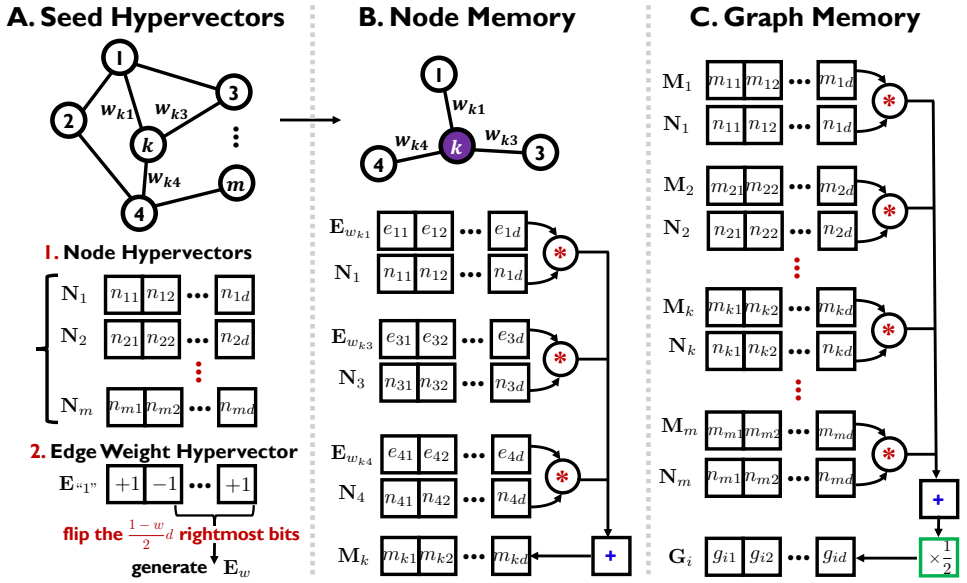


Fig. 3. Graph encoding for weighted undirected graphs in GrapHD [12].

TABLE I
CLASSIFICATION PERFORMANCE FOR THREE CLASSIFICATION TASKS

Approach	Emotion *vs.* Gambling				Emotion *vs.* No-task				Gambling *vs.* No-task			
Metrics[a]	Acc	Sen.	Spec.	AUC	Acc	Sen.	Spec.	AUC	Acc	Sen.	Spec.	AUC
GrapHD	75.48%	75.16%	75.80%	0.75	78.87%	78.98%	78.77%	0.79	80.57%	78.77%	82.38%	0.81
Record-based	75.27%	73.25%	77.28%	0.75	79.09%	79.62%	78.56%	0.79	80.04%	78.98%	81.10%	0.80
Linear SVM	94.27%	93.21%	95.33%	0.94	98.09%	97.66%	98.51%	0.98	97.88%	97.88%	97.88%	0.98

[a]In terms of metrics, "Acc" represents test accuracy, "Sen." is short for sensitivity, specificity is denoted by "Spec.".

under the ROC curve (AUC), where ROC refers to the receiver operating characteristic.

B. Performance Comparison

Table I demonstrates the simulation results for the two employed HDC encoding approaches given the same features—85×85 correlation-coefficient matrix. Additionally, the results of the traditional linear SVM approach are also considered as a reference. From this table, we observe that: *1)* with the same correlation-coefficient matrices as the features, the GrapHD achieves similar performance as the record-based encoding. *2)* The performance parameters of HDC encoding approaches are lower than the traditional linear SVM approach. This indicates the parameter-tuning for the hyperplane of SVM functions can separate the data quite well; however, HDC encoding approaches cannot perform as well as SVM in data separability for fully connected brain graphs. Therefore, sub-graphs are generated to further improve the HDC performance by utilizing sparsity.

In terms of HDC encoding approaches, GrapHD requires less memory storage than record-based encoding. As shown in Table II, for an arbitrary $m \times m$ correlation-coefficient matrix as the input feature, the memory requirement is $(m+1)$ for GrapHD and $\binom{m}{2}+1$ for record-based encoding. More specifically, for this classification problem using $m = 85$, record-based encoding requires $\sim 41\times$ memory storage as compared to GrapHD. Due to this reason, GrapHD is preferred.

TABLE II
MEMORY STORAGE[a] COMPARISON FOR TWO HD ENCODING APPROACHES

Approach	GrapHD	Record-based
Seed Hypervector Storage	$m+1$	$\binom{m}{2}+1$
	86	3571

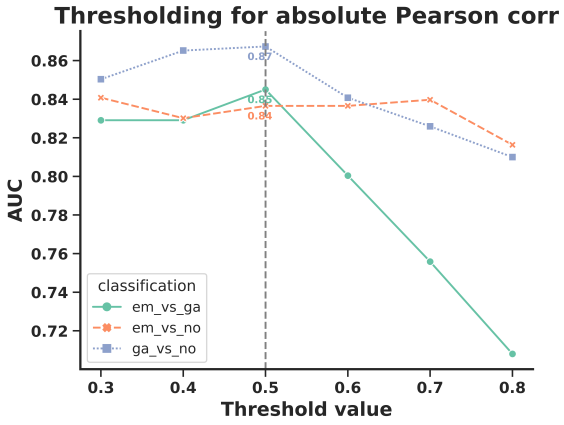
[a]Here the memory storage is defined by the number of hypervectors ($d = 10,000$).

C. Performance Improvement By Sub-Graphs

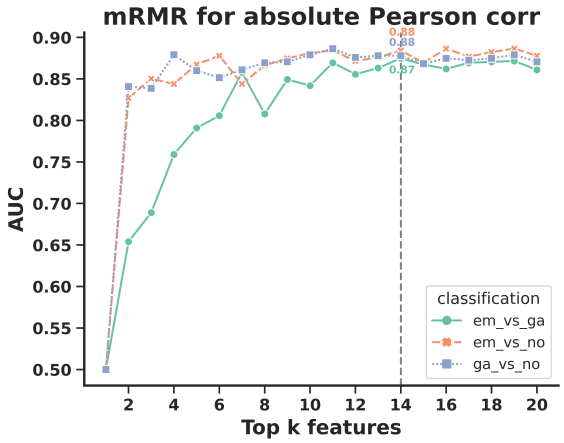
Instead of fully connected brain graphs, sub-graphs can improve the performance of binary classification using HDC. To utilize the sparsity of graphs, both the threshold strategy and mRMR algorithm are employed.

1) Threshold Strategy: For the given fully connected brain graphs, there will be no edge connection if the edge weight w is below a pre-defined threshold value Th. Here, this threshold value Th is tested from 0.3 to 0.8 with a step size of 0.1. As shown in Fig. 4(a), the $Th = 0.5$ leads to the optimal/suboptimal performance for all three binary classification problems. The corresponding AUCs are 0.85 of emotion *vs.* gambling, 0.84 of emotion *vs.* no-task, and 0.87 of gambling *vs.* no-task, respectively.

2) mRMR Algorithm: To generate sub-graphs, feature selection methods can be applied to select important edge weights from the fully connected graphs. Here, the mRMR algorithm is employed to select the top k features, where $k \leq 20$.



(a) Threshold strategy.



(b) mRMR algorithm.

Fig. 4. Performance improvement for GrapHD encoding by two strategies.

As shown in Fig. 4(b), the optimal/suboptimal performance for all three classification problems is achieved when $k = 14$. The corresponding AUCs are 0.87 of emotion *vs.* gambling, 0.88 of emotion *vs.* no-task, and 0.88 of gambling *vs.* no-task, respectively. These AUC results are significantly better than those with fully-connected brain graphs.

V. Conclusion

This paper investigates the application of HDC to brain graph classification using three different fMRI data. Both the record-based encoding and GrapHD are explored. Experimental results show that these two HDC approaches cannot achieve comparable classification performance with the traditional linear SVM. In terms of HDC encoding approaches, GrapHD requires significantly less memory storage than record-based encoding. Compared to fully connected brain graphs, the utilization of sparsity improves the classification performance of GrapHD encoding. This paper has not explored the use of retraining. Whether retraining can further improve the performance of HDC needs to be investigated. Recent work has shown that directed graphs based on causal information such as directed information can lead to higher accuracy in traditional classification. Future work should also be directed towards HDC classification based on directed brain graphs.

References

[1] P. Kanerva, *Sparse Distributed Memory*. MIT press, 1988.

[2] R. W. Gayler, "Vector symbolic architectures answer jackendoff's challenges for cognitive neuroscience," *arXiv preprint cs/0412059*, 2004.

[3] L. Ge and K. K. Parhi, "Classification using Hyperdimensional computing: A review," *IEEE Circuits and Systems Magazine*, vol. 20, no. 2, pp. 30–47, 2020.

[4] A. Joshi, J. T. Halseth, and P. Kanerva, "Language geometry using random indexing," in *International Symposium on Quantum Interaction*, Springer, 2016, pp. 265–274.

[5] M. Imani, D. Kong, A. Rahimi, and T. Rosing, "VoiceHD: Hyperdimensional computing for efficient speech recognition," in *2017 IEEE International Conference on Rebooting Computing (ICRC)*, IEEE, 2017, pp. 1–8.

[6] D. Kleyko, E. Osipov, A. Senior, A. I. Khan, and Y. A. Şekerciogğlu, "Holographic graph neuron: A bioinspired architecture for pattern processing," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 6, pp. 1250–1262, 2016.

[7] M. Imani, T. Nassar, A. Rahimi, and T. Rosing, "HDNA: Energy-efficient DNA sequencing using hyperdimensional computing," in *2018 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*, IEEE, 2018, pp. 271–274.

[8] A. Rahimi, P. Kanerva, L. Benini, and J. M. Rabaey, "Efficient biosignal processing using hyperdimensional computing: Network templates for combined learning and classification of ExG signals," *Proceedings of the IEEE*, vol. 107, no. 1, pp. 123–143, 2018.

[9] B. Sen, S.-H. Chu, and K. K. Parhi, "Ranking regions, edges and classifying tasks in functional brain graphs by sub-graph entropy," *Scientific reports*, vol. 9, no. 1, pp. 1–20, 2019.

[10] *Connectomedb database*, https://db.humanconnectome.org..

[11] A. Burrello, K. Schindler, L. Benini, and A. Rahimi, "One-shot learning for iEEG seizure detection using end-to-end binary operations: Local binary patterns with hyperdimensional computing," in *2018 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, IEEE, 2018, pp. 1–4.

[12] P. Poduval, A. Zakeri, F. Imani, H. Alimohamadi, and M. Imani, "Graphd: Graph-based hyperdimensional memorization for brain-like cognitive learning," *Frontiers in Neuroscience*, p. 5, 2022.

[13] M. Heddes, I. Nunes, P. Vergés, D. Desai, T. Givargis, and A. Nicolau, "Torchhd: An open-source python library to support hyperdimensional computing research," *arXiv preprint arXiv:2205.09208*, 2022.

[14] I. Nunes, M. Heddes, T. Givargis, and A. Nicolau, "An extension to basis-hypervectors for learning from circular data in hyperdimensional computing," *arXiv preprint arXiv:2205.07920*, 2022.

[15] E. P. Frady, D. Kleyko, and F. T. Sommer, "Variable binding for sparse distributed representations: Theory and applications," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[16] L. Ge and K. K. Parhi, "Applicability of hyperdimensional computing to seizure detection," *IEEE Open Journal of Circuits and Systems*, vol. 3, pp. 59–71, 2022.