Hyperdimensional Computing based Classification and Clustering: Applications to Neuropsychiatric Disorders

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Lulu Ge

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Prof. Keshab K. Parhi, Advisor

December, 2023

Acknowledgements

I would like to thank my advisor Prof. Keshab K. Parhi for his consistent guidance and continued support throughout my whole Ph.D. journey. My gratitude goes out to Prof. Alik Widge, my mentor, for his guidance and assistance during the Institute for Engineering in Medicine's (IEM) research project.

I would like to express my gratitude towards the members of my committee, namely Prof. Chris Kim, Prof. Marc Riedel, and Prof. Ru-yu Lai (for my preliminary oral exam).

I would like to thank all my lab colleagues at Parhi's lab for both research discussion and mental support. My gratitude goes out to Sandeep Avvaru and Nanda Kumar Unikrishnan for patiently answering my questions, encouraging me to overcome my fears, and validating that I have made progress. My sincere thanks go to Xingyi Liu for his discussions to improve my research thoughts and for his conversation in Chinese to ease my homesickness. I would like to extend a special thank you to Sai Sanjay Balaji for his patience in improving my English comprehension and encouraging me to take action rather than overthinking. I would like to express my gratitude to Sai Sanjeet Yerraguntala and Joe Gould to discuss my research. My Ph.D. journey would not have been possible without their help and support.

I am grateful to Aaron McInnes for assisting me with my IEM project. His support in data collection, result analysis, and guidance on classical algorithms in R language was truly valuable.

I owe a debt of gratitude to many other friends, both within and outside the US, who have supported me over the years. I am grateful for the unwavering support provided by Anlan Yu and Shusen Jing throughout my Ph.D. journey. I would like to extend a special thank you to Wenqing Song for her encouragement. My gratitude goes to

Zhizhen Wu for his mental support. I would like to thank Yangyang Chang, Vasanth Ravikumar, Ziyuan Shen, Yuxiang Lu, Weihong Xu, Chouzhou Fang, Zhongqi Zhao, Ziqing Zeng, Huan Liu, Yifei Shen, Xufeng Bao, Si Shen, Baozhen Wang, Yuhan Guo, Pengguang Li, Xiaopeng Li and Jie Deng for their continued support and belief in me. I would like to express my heartfelt gratitude to my friends at Wayzata Church for welcoming me into their community and making me feel cared for. My sincere thanks go out to Sai Wang for her care and support. I am especially grateful to my host family, Bob and Bonnie, who provided me with beautiful memories during my time in Minnesota. I will always cherish our time together and look forward to keeping in touch.

Most importantly, I would like to thank my family, especially my parents, for their love, understanding, and patience. Without their steadfast support, I would not have been able to achieve this milestone in my life.

Dedication

This dissertation is dedicated to my grandparents, Baogong Ge, and Jinhua Dai, who supported my decision to study abroad instead of marrying early. Although they passed away shortly after I arrived in the US to pursue my Ph.D. degree, their unconditional love has continued to support me throughout my academic journey. Their unwavering belief in my abilities and their encouragement to pursue my dreams have been a tremendous source of strength. I am grateful for the time we had together, and I will always cherish the memories of their kindness, wisdom, and generosity. This work is a tribute to their memory, and I hope it would have made them proud.

Abstract

Since its introduction in 1988, hyperdimensional computing (HDC), also referred to as vector symbolic architecture (VSA), has attracted significant attention. Using hypervectors as unique data points, this brain-inspired computational paradigm represents, transforms, and interprets data effectively. So far, the potential of HDC has been demonstrated: comparable performance to traditional machine learning techniques, high noise immunity, massive parallelism, high energy efficiency, fast learning/inference speed, one-/few-shot learning ability, etc. In spite of HDC's wide range of potential applications, relatively few studies have been conducted to demonstrate its applicability. To this end, this dissertation focuses on the application of HDC to neuropsychiatric disorders: (a) seizure detection and prediction, (b) brain graph classification, and (c) transcranial magnetic stimulation (TMS) treatment analysis. We also develop novel clustering algorithms using HDC that are more robust than the existing HDCluster algorithm.

In order to detect and predict seizures, intracranial electroencephalography (iEEG) data are analyzed through the use of HDC-based local binary pattern (LBP) and power spectral density (PSD) encoding. Our study examines the effectiveness of utilizing all features as well as a small number of selected features. Our results indicate that HDC can be used for seizure detection, where PSD encoding is superior to LBP encoding. We observe that even three features are efficient in detecting seizures with PSD encoding. However, in order to pave the way for seizure prediction using HDC, more efficient features must be explored.

For the classification of brain graphs, data from functional magnetic resonance imaging (fMRI) are analyzed. Brain graphs describe the functional brain connectome under varying brain states, and are generated from the fMRI data collected at rest and during tasks. The brain graph structure is assumed to vary from task to task and from task to no task. Participants are asked to execute emotional and gambling tasks, but no tasks are assigned during resting periods. GrapHD, an HDC-based graph representation, initially developed for object detection, is herein expanded for the application to brain graph classification. Experimental results demonstrate that GrapHD encoding has the capability of classifying the brain graphs for three binary classification problems:

emotion *vs.* gambling, emotion *vs.* no-task, and gambling *vs.* no-task. Furthermore, GrapHD requires fewer memory resources as compared to the extant HDC-based encoding approaches.

In terms of clustering, HDCluster, an HDC-based clustering algorithm, has been proposed in 2019. Originally designed to mimic the traditional k-means, HDCluster exhibits higher clustering performance across versatile datasets. However, we have identified that the performance of the HDCluster may be significantly influenced by the random seed used to generate the seed hypervectors. To mitigate the impact of this random seed, we propose more robust HDC-based clustering algorithms, designed to outperform HDCluster. Experimental results substantiate that our HDC-based algorithms are more robust and capable of achieving higher clustering performance than the HDCluster.

In the analysis of TMS treatment, we conduct two specific tasks. One is to identify the clinical trajectory patterns for patients who suffer from major depressive disorder (MDD) (Clustering). Another is to predict MDD severity using 34 measured cognitive variables (Classification). For clustering, we propose a novel HDC-based algorithm that manipulates HDCluster to determine the number of clusters for a system of clinical trajectories. For classification, we utilize two HDC-based encoding algorithms and examine the impact of using either all features or selected features. Experimental results indicate that our HDC algorithm mirrors the clustering pattern of the classical algorithm. Additionally, the HDC-based classifier effectively predicts the concept of clinical response.

Contents

Acknowledgements **i**

Dedication **iii**

Abstract **iv**

Contents **vi**

List of Tables **x**

List of Figures **xii**

1 Introduction **1**

1.1 Research Overview 1

1.2 Dissertation Outline 4

1.3 Summary of Contributions 6

2 Classification using HDC: a review **8**

2.1 Introduction 8

2.2 Background on HDC 9

2.2.1 Classical Computing vs HDC 9

2.2.2 Data Representation 10

2.2.3 Similarity Measurement 11

2.2.4 Data Transformation 13

2.3 Learning and Classification By HDC 17

2.3.1 The HD Classification Methodology 18

2.3.2 Encoding Methods for HDC 18
2.3.3 Benchmarking Metrics in HDC 21
2.4 Applications in HD Classification 23
2.4.1 Letters 23
2.4.2 Signals 25
2.4.3 Images 32
2.4.4 Summary 33
2.5 Conclusion 36

3 Applicability of seizure detection using HDC 37
3.1 Introduction 37
3.2 Methodology 39
3.2.1 LBP Method 39
3.2.2 PSD Method 41
3.2.3 Hypervector Distance Plot 45
3.3 Materials 46
3.3.1 iEEG Dataset from Kaggle Contest 46
3.3.2 Training, Validation and Test Data 47
3.3.3 Performance Evaluation 47
3.4 Experimental Results 48
3.4.1 LBP Method 48
3.4.2 PSD Method 50
3.4.3 Discussion on LBP and PSD Methods 53
3.5 Conclusion 57

4 Applicability of seizure prediction using HDC 60
4.1 Introduction 60
4.2 Preliminaries 61
4.2.1 Basics of HDC 61
4.2.2 Seizure Prediction Dataset 61
4.2.3 Flow Chart of the Employed Approaches 62
4.2.4 Training and Test Workflow 62
4.3 Methodology 63

4.3.1 LBP Method 63
4.3.2 PSD Method 65
4.4 Experimental Results 67
4.5 Conclusions 68

5 Classifying functional brain graphs using graph hypervector representation 69

5.1 Introduction 69
5.2 Methodology 71
5.2.1 Record-based Encoding 71
5.2.2 GrapHD Encoding 72
5.3 Materials 73
5.3.1 HCP Dataset and Preprocessing 73
5.3.2 Flowchart of The Approaches 74
5.4 Experimental Results 74
5.4.1 Training and Test Workflow 74
5.4.2 Evaluation Metrics 75
5.4.3 Performance Comparison 75
5.4.4 Performance Improvement By Sub-Graphs 76
5.5 Conclusion 77

6 Clustering using HDC 78

6.1 Introduction 78
6.1.1 Traditional Clustering Algorithms 80
6.1.2 HDCluster 81
6.2 Methodology 82
6.2.1 Similarity-Based K-means 83
6.2.2 Equal Bin-Width Histogram 83
6.2.3 Equal Bin-Height Histogram 84
6.2.4 Similarity-Based Affinity Propagation 85
6.3 Materials 85
6.3.1 Dataset Description 85
6.4 Experimental Results 86

6.4.1 Experimental Setup 86
6.4.2 Comparison with HDCluster 86
6.4.3 Original Data Space vs Hyperdimensional Space 92
6.4.4 Further Discussion 93
6.5 Conclusion 96

7 Clustering and classification for brain stimulation with HDC 97

7.1 Introduction 97
7.2 Preliminaries 100
7.2.1 Basics of HDC 100
7.2.2 Classification using HDC 102
7.3 Methodology 103
7.3.1 Clustering using HDC 103
7.3.2 Category Prediction using HDC 104
7.4 Experimental Results 106
7.4.1 Materials 106
7.4.2 Clustering 106
7.4.3 Category Prediction 112
7.4.4 Further Discussion 113
7.5 Conclusion 114

8 Conclusion and Future Directions 116

8.1 Key Findings 116
8.2 Future Outlook 117

References 119

Appendix A. Supplementary Results 137

A.1 Clustering Results using HDC. 137
A.1.1 Small Dataset 137
A.1.2 Large Dataset 137
A.2 Clustering Results using LCMM. 140
A.2.1 Large Dataset 140

List of Tables

2.1 Comparisons between classical computing and HDC for classification. . 10
2.2 Similarity measurements in HDC. 11
2.3 Summary of the strategies used in HDC for accuracy and efficiency improvement. 33
2.4 Partial list of applications based on HDC in [1]. 34
3.1 Three selected features based on Fisher score. 44
3.2 Dataset information. 47
3.3 Performance metrics. 48
3.4 Average LBP validation AUC performances. 49
3.5 Performance for selected PSD features using fisher score with quantization level $q = 21$.[1] 50
3.6 Performance for all PSD features with quantization level $q = 21$.[1,2] . . . 51
3.7 Summary for LBP and PSD methods. 51
3.8 Impact of hypervector sizes for the test AUC using LBP and PSF methods. 52
3.9 Memory and computational requirements for LBP and PSD methods. . 52
3.10 Specific memory requirements for LBP and PSD methods. 53
4.1 Dataset information. 62
4.2 Experimental results for LBP method with the code length $l = 6$. 67
4.3 Experimental results for PSD method with the quantization level $q = 21$. 68
5.1 Classification performance for three classification tasks. 75
5.2 Memory storage[1] comparison for two HD encoding approaches. 76
6.1 Datasets for clustering using HDC. 85
6.2 Performance for three traditional clustering algorithms[1]. 93

6.3 Performance comparison of our proposed algorithms and the baseline HDCluster[1]. 95
7.1 LCMM results for original data ($n = 27$). 109
7.2 LCMM results for corrected data ($n = 27$). 110
7.3 Experimental results for category prediction using HDC. 111
7.4 Comparison with SVM. 113
A.1 LCMM results for original data ($n = 176$). 140
A.2 LCMM results for corrected data ($n = 176$). 141

List of Figures

1.1 Research tasks outlined in this dissertation. 3
1.2 Dissertation outline. 5
2.1 Orthogonality in high dimensions [2–4]. 12
2.2 Hamming distance distribution of addition for 10,000-bit hypervectors over 3000 cases. (a) Addition over an odd number of hypervectors; (b) and (c) shows the addition over even number favoring 0 and 1, respectively. 13
2.3 Hamming distance distribution of multiplication $\mathbf{X} = \mathbf{A} \oplus \mathbf{B}$ for 10,000-bit hypervectors over 3000 cases. 15
2.4 Hamming distance distribution of permutation for 10,000-bit hypervectors over 3000 cases. 16
2.5 Classification overview with HDC [5]. 18
2.6 Record-based encoding [6]. Note iM refers to item memory, which stores the position hypervectors, and CiM refers to continuous item memory [3], which stores level hypervectors. 19
2.7 N-gram-based encoding [7]. CiM stores level hypervectors that are mutually orthogonal. 20
2.8 Two benchmarking metrics in HDC and some possible ways to improve these metrics. 21
2.9 The architecture for language recognition with HDC [1, 5]. 23
2.10 VoiceHD+NN flow for training and testing [8]. 25
2.11 The architecture for Laelaps with HDC to detect and alarm seizure [9]. 25
2.12 CompHD for (a) an HD model and (b) a query data [10]. 28
2.13 Overview of SemiHD framework supporting self-training in HD space [11]. 29
2.14 Block diagram of the HD Character Recognition System [12]. 31

3.1 HD classification using LBP method. 40
3.2 HD classification using PSD method. 42
3.3 A general hypervector distance plot for an ideal binary classification. . . 46
3.4 AUC for validation data. 49
3.5 Hypervector distance plots for Patient_2 and Dog_4 among different HD classifiers. 57
4.1 Flow chart of the employed approaches. 63
4.2 HDC classification using LBP method. 64
4.3 HDC-based PSD method for seizure prediction. 65
5.1 Graph encoding for weighted undirected graphs in GrapHD [13]. 72
5.2 Flowchart of the two HDC approaches for the brain state classification. 74
5.3 Performance improvement for GrapHD encoding by two strategies. . . . 76
6.1 HDCluster overview [14]. In [14], the encoder of the original HDCluster refers to record-based encoding, and the initial cluster centers are random hypervectors. 82
6.2 Proposed HDC-based clustering algorithms. 83
6.3 Examples for the equal bin-width histogram. 84
6.4 Examples for the equal bin-height histogram. 85
6.5 Comparison of proposed algorithms with HDCluster. For boxplots, the median values over 500 runs/trials are annotated on the top. The red values indicate the results for the baseline HDCluster. Our algorithms' results are in black. The symbol × implies the results are not available. 88
6.6 Comparison of one-pass clustering with both the updated clusters and HDCluster. For boxplots, the median values over 500 runs/trials are annotated on the top. The red values indicate the results for the baseline HDCluster. Our algorithms' results are in black. The symbol × implies the results are not available. 90
6.7 Comparison of clustering using original data and encoded data. For boxplots, the median values over 500 runs/trials are annotated on the top. Results for encoded data are in red, while those for the original data are in black. 91
7.1 Goal overview. 98

7.2 Overview of the basics of HDC. 100
7.3 Clinical-trajectory-pattern clustering using HDC. 103
7.4 Category prediction overview. 105
7.5 Clinical-trajectory-pattern clustering using HDC for a small dataset ($n = 27$). 107
7.6 Clinical-trajectory-pattern clustering using HDC for a large dataset ($n = 176$). 108
7.7 Trajectory clustering for LCMM when $n = 27$. 111
A.1 Clinical-trajectory-pattern clustering using HDC for a small dataset ($n = 27$). 138
A.2 Clinical-trajectory-pattern clustering using HDC for a large dataset ($n = 176$). 139
A.3 Trajectory clustering for LCMM when $n = 176$. 141

Chapter 1

Introduction

1.1 Research Overview

Hyperdimensional computing (HDC), a.k.a vector symbolic architecture (VSA), is a novel computing paradigm that draws inspiration from the structure and functionality of the human brain. Unlike traditional computing, which operates on binary bits, HDC operates on *hypervectors* (ultra-long vectors), enabling it to perform complex computations with remarkable efficiency and accuracy. While HDC is a relatively new field, researchers have reported promising results in a wide range of applications, from machine learning and artificial intelligence to robotics and neuroscience.

HDC has a rich history in the field of cognitive computing and artificial intelligence. HDC was introduced by Pentti Kanerva in the 1980s [2]. Kanerva's work was influenced by earlier ideas in distributed representation and high-dimensional (HD) spaces. HDC has evolved over the years, with researchers expanding on the original concepts and applying them to various domains, including natural language processing, cognitive modeling, and machine learning. HDC has found applications in cognitive science, robotics, machine learning, and neural network research. It is known for its ability to represent and manipulate complex data in high-dimensional spaces.

HDC is known for the following main strengths:

- **Robustness to noise.** The distributed representation of HDC enables it to encode information across all dimensions of hypervectors. This means that even if there are errors or noise in a few positions, the overall quality of the encoding is not significantly affected [15].

- **Efficiency.** Unlike convolutional neural networks (CNNs) and deep learning (DL) models that rely on backpropagation, HDC involves only bitwise addition, multiplication, and permutation operations, making it suitable for hardware implementation, and in-memory computing.
- **Model interpretation.** By representing data as hypervectors, HDC can identify patterns and relationships that may not be immediately apparent in lower-dimensional representations.
- **Scalability.** In the context of HDC, scalability refers to how well HDC can adapt to larger datasets, higher-dimensional spaces, and more complex tasks. Since HDC is inherently designed to work in HD space, this feature makes it highly scalable when dealing with data that has a large number of features or dimensions. As the volume of data grows, HDC can handle the increased data without a significant increase in computational complexity. Its memory-efficient representations and ability to operate on subsets of data make it suitable for big data applications.
- **Few-shot learning.** HDC can learn from a few samples but achieve great performance for certain applications [16]. This few-shot learning capability of HDC offers a promising solution when training data is limited.
- **Interoperability.** HDC can integrate with existing machine learning and deep learning frameworks, allowing researchers and practitioners to incorporate its advantages into their existing workflows [8].
- **Parallelism.** Modern parallel computing architectures, such as multi-core processors and GPUs, enable HDC operations to be parallelized effectively. As a result of this parallelism feature, HDC is capable of processing large datasets at faster speeds.

While HDC has the aforementioned advantages, it also comes with some limitations and potential drawbacks: (a) **Lack of mainstream adoption.** The HDC approach is not widely used or accepted in the mainstream machine learning community or industry despite its potential and advantages. This highlights the need for continued efforts to identify and develop the “killer application” of HDC that can demonstrate its full potential. (b) **Application specificity.** HDC may be better suited for certain types of tasks, such as language recognition [17] and DNA sequencing [18], and may not be the best choice for all machine learning and data analysis problems. Pilot studies are necessary to evaluate the suitability of HDC for a particular task. (c) **Encoding complexity.** While HDC can capture complex relationships, encoding specific patterns or structured information may require careful design of the encoding scheme.

In light of the limitations stated above, this dissertation investigates the feasibility of HDC for specific tasks. The focus of this research is on the application of HDC to neuropsychiatric disorders, including seizure detection and prediction using intracranial electroencephalogram (iEEG) data, brain graph classification using functional magnetic resonance imaging (fMRI) data, and transcranial magnetic stimulation (TMS) treatment analysis using brain stimulation measurement data. These tasks involve classification and clustering, which are critical to the diagnosis and treatment of neurological disorders. The research examines HDC’s applicability and evaluates its effectiveness and efficiency in solving real-world biosignal analysis problems.

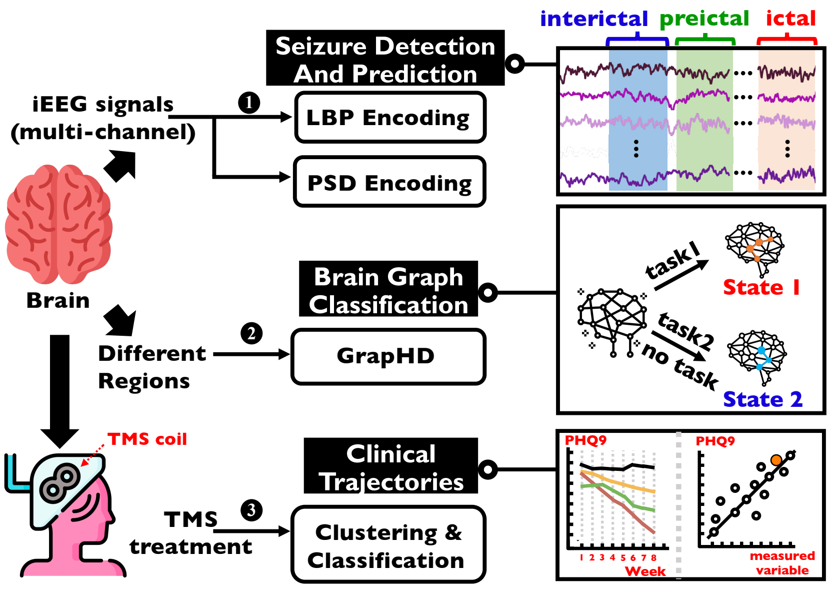


Figure 1.1: Research tasks outlined in this dissertation.

Figure 1.1 vividly shows three research tasks that are covered in the dissertation: (a) **(Seizure detection and prediction)** A binary classification problem based on multi-channel iEEG data aimed at identifying brain activity. There is an imbalance in the data when it comes to seizure detection (*interictal vs. ictal*) and seizure prediction (*interictal vs. preictal*). (b) **(Brain graph classification)** Brain connectivity varies depending on the task being performed, both between different tasks (e.g., Task A *vs.* Task B), as well as between task and no-task conditions. The brain graphs, consisting of nodes and edges, are generated by looking at correlation coefficients between functional brain regions/nodes captured by fMRI. Three binary classification problems are studied—emotion *vs.* gambling, emotion *vs.* no-task, and gambling *vs.* no-task. (c) **(TMS treatment analysis)** Individuals diagnosed with major depressive disorder (MDD) may benefit from repetitive transcranial magnetic stimulation (rTMS) treatment, which typically lasts for several weeks. Two specific aims are for the TMS treatment analysis in this dissertation. Identifying the clinical trajectory patterns for TMS treatment (Aim 1). Category prediction using measured cognitive variables (Aim 2). Note that all of the aforementioned data are collected from brains using different measurements or devices.

One thing that should be emphasized is that the research presented in this dissertation is algorithmic-oriented and focuses primarily on investigating the applicability of HDC to specific biosignal diagnosis tasks and how to encode/tailor HDC to these tasks. Hardware implementation is out of this research scope. Overall, this dissertation aims to contribute to the growing body of research on HDC and its potential applications in neuroscience and medicine.

1.2 Dissertation Outline

As illustrated in Figure 1.2, the dissertation is organized as follows:

- Chapter 2 provides the basics of HDC, including the data representation, transformation, and interpretation. This chapter also presents a comprehensive review of classification using HDC.
- In Chapter 3, two encoding algorithms are introduced for seizure detection. This chapter investigates the applicability of seizure detection using HDC.

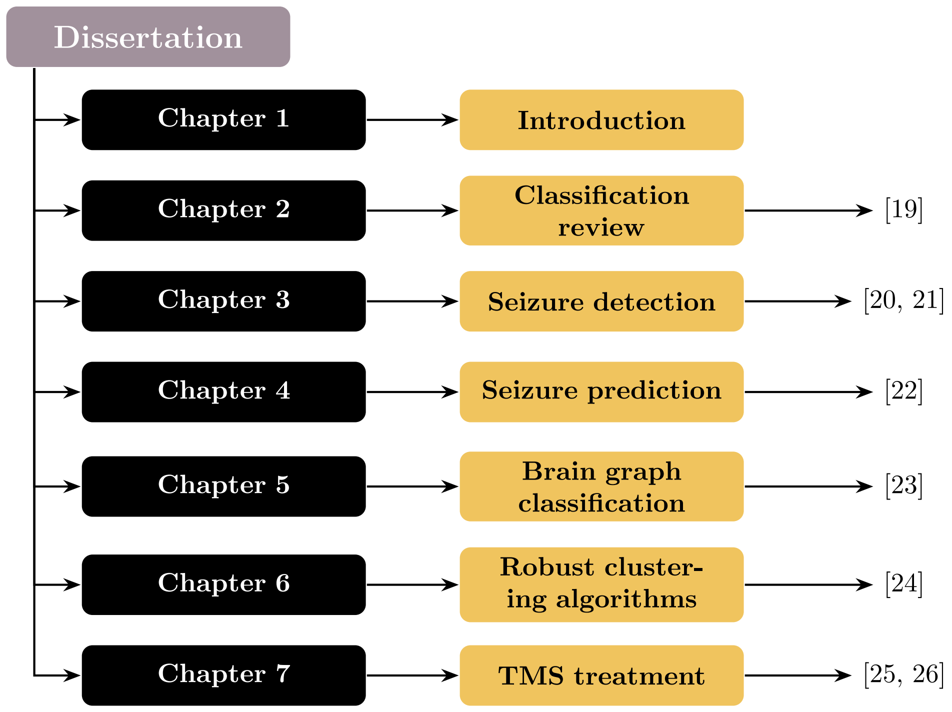


Figure 1.2: Dissertation outline.

- Chapter 4 tailors the two encoding algorithms presented in Chapter 3 for seizure prediction. The applicability of seizure prediction using HDC is investigated in this chapter.

- Chapter 5 describes the specific research task for brain graph classification. A novel graph representation, GrapHD encoding [13], is applied to brain graph classification.

- Chapter 6 reviews the existing HDC-based clustering algorithm, HDCluster [14]. This chapter also illustrates our proposed HDC-based clustering algorithms, which are more robust than HDCluster.

- In Chapter 7, two specific research tasks for TMS treatment analysis are described. The first task involves utilizing the existing HDCluster to determine the optimal number of clusters for a given system. The second task offers a new perspective on prediction accuracy, suggesting that predicting categories rather than numerical values may be more effective.

- A complete summary of key results and a brief description of future work are included in the conclusive Chapter 8.

Overall, this dissertation aims to evaluate the effectiveness and efficiency of HDC for specific biosignal-centric tasks. By addressing these research questions, this study provides insights into the potential applications and limitations of HDC for real-world biosignal analysis problems.

1.3 Summary of Contributions

The contributions of this research can be outlined as follows.

- We recognized the potential of HDC early on, at a time when it had not yet gained widespread attention from researchers. We have since conducted extensive research on HDC and have summarized the most recent advancements in this area, particularly in the context of classification using HDC. Our work provides a valuable resource for new researchers who are interested in exploring the potential of HDC and need to quickly understand its basics. By highlighting the fundamental concepts and latest research progress in HDC, this dissertation helps to bridge the gap between theory and practice and to promote further research in this area.

- Previous research has shown that HDC can effectively solve short-term [16] and long-term [9] seizure detection using the SWEC-ETHZ iEEG database [27]. To further validate this finding and test the generalizability of HDC, we investigate its applicability for seizure detection using the Kaggle dataset [28]. Unlike prior work [9, 16] where time-domain features (e.g., local binary patterns (LBP)) are employed, we adopt frequency-domain features—power spectral density (PSD) features. Additionally, both all features and a number of selected features are studied for seizure detection using HDC classifiers.

- Seizure prediction is a complex task that requires distinguishing between interictal and preictal states, which can be challenging due to the absence of clear differences between them. In this dissertation, we aim to address this difficulty by exploring new approaches to seizure prediction. Our research focuses on the use of HDC for seizure prediction, and while we are unable to demonstrate its applicability using the Kaggle prediction dataset, we can determine that both LBP and PSD features are not effective for seizure prediction. These findings represent an important contribution to the field of seizure prediction and highlight the need for continued research into more effective methods for predicting seizures.

- We explore the potential of the GrapHD encoding algorithm [13], which is a novel data representation to encode graph information that consists of nodes and edges. While GrapHD was not developed by us, we applied this innovative technique to the task of brain graph classification. We found it to be highly effective in memory resources as compared to the existing encoding approaches in HDC.

- Although HDCluster [14] has shown higher clustering performance than traditional k-means across diverse datasets, we found that the clustering results are heavily influenced by the random seeds used to generate the seed hypervectors. This finding highlights the importance of considering the random seed in parameter tuning for performance analysis. To address this issue, we propose more robust HDC-based clustering algorithms that are less sensitive to the choice of random seed. Our results demonstrate that these new algorithms outperform HDCluster, providing a more reliable and effective approach to clustering with HDC.

- We explore the application of HDC to TMS treatment analysis, making two significant contributions. Firstly, we propose a novel algorithm that utilizes HDCluster to determine the optimal number of clusters for a given TMS treatment system, allowing for the identification of clustering patterns in clinical responses. Secondly, we address the challenge of predicting actual numerical values for clinical purposes and instead propose a pipeline that focuses on category prediction, showing high performance in accurately classifying different categories related to TMS treatment outcomes.

Chapter 2

Classification using HDC: a review

Research for the following chapter has been published in [19]. In this chapter, a detailed overview of HDC is provided, including its theoretical background. Specifically, this chapter presents a summary of classification using HDC.

2.1 Introduction

The emergence of HDC is based on the cognitive model developed by Kanerva [2]. HDC grew out of cognitive science in answer to the binding problem of connectionist (neural-net) models. When variables and their values are superposed over the same vector, representing which value is associated with which variable requires a formal model. This was initially solved using tensor product variable binding by Smolensky [29] and later by Plate[30] using holographic reduced representation (HRR). The advantage of HRR over tensor product is that it keeps vector dimensionality constant. Systems based on these representations go by many names: HRR, HD, binary spatter code (BSD) [31], binary sparse distributed code (BSDC) [32], multiply-add-permute (MAP) [33], vector symbolic architecture (VSA) [34], and semantic pointer architecture. All rely on high dimensionality, randomness, abundance of nearly orthogonal vectors, and computing in superposition.

Instead of computing traditional numerical values, HDC performs cognition tasks—such as face detection, language classification, speech recognition, image classification, etc—by representing different types of data using hypervectors, whose dimensionality is in the thousands, e.g., 10,000-d, where d refers to dimensionality. The human brain contains about 100 billion neurons and 1000 trillion synapses; therefore all possible *states* of a human brain can be described by a high-dimensional vector. In that sense, HDC is a form of brain-inspired computing. Randomly or pseudo-randomly defined, these hypervectors are composed of independent and identically distributed (*i.i.d.*) components, which can be binary, integer, real or complex [35]. As a brain-inspired computing model, HDC is robust, scalable, energy efficient and requires less time for training and inference [36]. These features are a result of its ultra-wide data representation and underlying mathematical operations. One thing that should be emphasized is the concept of *orthogonality* of the hypervectors.

2.2 Background on HDC

In this section, we review HDC and present a comparison between HDC and classical computing. We also describe the similarity metrics for hypervectors and typical mathematical operations used in HDC.

2.2.1 Classical Computing vs HDC

Data representation, data transformation, and data retrieval play an important role in any computing system. To be more specific, classical computing deals with bits. Each bit is 0 or 1. This can be realized by the absence or presence of an electric charge. In terms of computation, data transformation is inevitable. The arithmetic/logic unit (ALU) computes new data using logical operation and four arithmetic operations, including addition, subtraction, multiplication, and division [37]. The main memory allows the data to be written and read. Compared to classical computing, HDC employs hypervectors as its data type, whose dimensionality is typically in the thousands. These ultra-wide words introduce redundancy against noise, and are, therefore, inherently robust.

To transform data, HDC performs three operations: multiplication, addition, and

Table 2.1: Comparisons between classical computing and HDC for classification.

Computing Paradigm	**Classical Computing**	**HDC**
Data Type	Bit	Hypervector
Data Transformation	Addition, Multiplication, Logic	Add-Multiply-Permute
Storage	Memory	Item Memory, Associative Memory
Training	Weights	Class Hypervectors
Testing	Run Pre-trained Classifier	Associate Query Hypervectors with Class Hypervectors
Model Complexity	High	Low
Accuracy	Very High	Acceptable
Feature Encoding	Easy	Difficult
Number of Features	Many	One

permutation. HDC transforms the input hypervectors, which are pre-stored in the item memory to form associations or connections. In a classification problem, the hypervectors associated with classes are trained during the training process. During the testing process, the test hypervectors are compared with the class hypervectors. The hypervectors generated from training data are referred to as *class* hypervectors and are stored in the associative memory, while those generated from the test data are referred to as *query* hypervectors. An associative search is performed to make a prediction as to which class a given query hypervector most likely belongs. A comparison between the classical and HDC paradigms is summarized in Table 2.1. Traditional classification methods achieve high accuracy using complex models. Training these models typically takes longer time and requires more energy consumption. The models in HDC-based classification are simpler and can be trained in less time with high energy efficiency. However, their accuracy is acceptable, though not as high as traditional models. This is because the accuracy is dependent on feature encoding which is not as well understood as traditional classification.

2.2.2 Data Representation

Data points of HDC correspond to hypervectors—vectors of bits, integers, real or complex numbers. These are roughly divided into two categories: binary and non-binary. For non-binary hypervectors, bipolar and integer hypervectors are more commonly employed. Generally speaking, non-binary HDC algorithms achieve higher accuracy, while the binary counterpart is more hardware-friendly and has higher efficiency (see also

[38]).

2.2.3 Similarity Measurement

As shown in Table 2.2, two common similarity measurements are adopted in the existing HDC algorithms, namely, cosine similarity and Hamming distance. Other similarity measures include dot product (e.g., in MAP) and overlap (e.g., in BSDC).

Table 2.2: Similarity measurements in HDC.

Measurement	Similar	Orthogonal
Hamming distance	0	0.5
Cosine similarity	1	0

For non-binary hypervectors, *cosine similarity*, defined by Eq. (2.1), is used to measure their similarity, focusing on the angle and ignoring the impact of the magnitude of hypervectors, where $|\cdot|$ denotes the magnitude. Unlike the inner product operation [39] of two vectors that affects magnitude and orientation, the *cosine similarity* only depends on the orientation. In most HDC algorithms with non-binary hypervectors, cosine similarity is more often used than inner product. Moreover, when $\cos(\mathbf{A}, \mathbf{B})$ is close to 1, this implies an extremely high level of similarity. For example, $\cos(\mathbf{A}, \mathbf{B}) = 1$ indicates two hypervectors $\mathbf{A}$ and $\mathbf{B}$ are identical. When they are at the right angle, then $\cos(\mathbf{A}, \mathbf{B}) = 0$, and the two orthogonal vectors are considered dissimilar.

$$\cos(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{|\mathbf{A}||\mathbf{B}|} \tag{2.1}$$

For binary hypervectors with dimensionality d, whose components are either 0 or 1, normalized Hamming distance calculated in Eq. (2.2) is used to measure their similarity [35]. When the Hamming distance of two hypervectors is close to 0, then they are defined as similar. For example, $\text{Ham}(\mathbf{A}, \mathbf{B}) = 0$ indicates every single bit is the same at each position, and $\mathbf{A}$ and $\mathbf{B}$ are identical. When $\text{Ham}(\mathbf{A}, \mathbf{B}) = 0.5$, $\mathbf{A}$ and $\mathbf{B}$ are orthogonal or dissimilar. $\text{Ham}(\mathbf{A}, \mathbf{B}) = 1$ when $\mathbf{A}$ and $\mathbf{B}$ are diametrically opposed.

$$\text{Ham}(\mathbf{A}, \mathbf{B}) = \frac{1}{d} \sum_{i=1}^{d} 1_{\mathbf{A}(i) \neq \mathbf{B}(i)} \tag{2.2}$$

In short, similarity measurement (δ) between hypervectors, can be summarized by Eq. (2.3), binary HDC uses the Hamming distance whereas non-binary HDC employs cosine similarity. Here $\mathbf{A}$, $\mathbf{B}$ are two hypervectors and d represents their corresponding dimensionality.

$$\delta(\mathbf{A}, \mathbf{B}) = \begin{cases} \frac{1}{d}\sum_{i=1}^{d} 1_{\mathbf{A}(i)\neq\mathbf{B}(i)}, \text{ binary HDC,} \\ \frac{\mathbf{A}\cdot\mathbf{B}}{|\mathbf{A}||\mathbf{B}|}, \text{ non-binary HDC.} \end{cases} \tag{2.3}$$

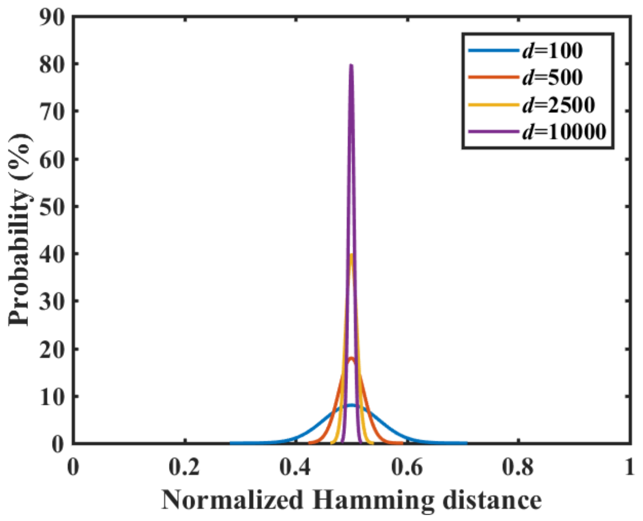
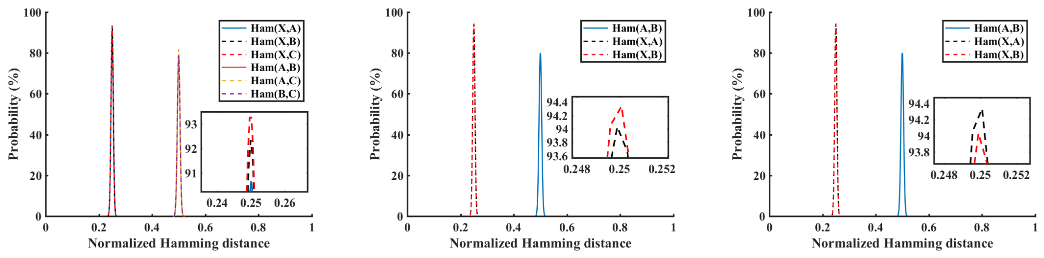


Figure 2.1: Orthogonality in high dimensions [2–4].

One thing that should be emphasized is orthogonality in high dimensions. To put it simply, the randomly generated hypervectors are nearly orthogonal to each other when the dimensionality is in the thousands. Take binary hypervectors as an example. Assume $\mathbf{X}$ and $\mathbf{Y}$ are chosen independently and uniformly from $\{0, 1\}^d$ and the probability p of any bit being 1 is 0.5. Then Ham($\mathbf{X}, \mathbf{Y}$) is binomially distributed. Figure 2.1 shows the probability density function (PDF) of Ham($\mathbf{X}, \mathbf{Y}$) for 15,000 pairs of randomly selected binary vectors with different dimensions d. As d increases, more vectors become orthogonal. Such orthogonality property is of great interest because orthogonal hypervectors are dissimilar. Moreover, operations performed on these orthogonal hypervectors can form associations or relations.

2.2.4 Data Transformation

Three types of operations, add-multiply-permute, are employed in HDC. The inverse operation for multiplication is also referred to as *release* [4]. The release operation is also used to denote inverse addition. Each operation processes and generates d-dimensional hypervectors. In the following, we illustrate examples of data transformations using binary hypervectors. Without doubt, data transformation can also be employed to non-binary hypervectors, which is in essence similar to the manipulations over binary hypervectors. The only difference is from the point of hardware; for binary hypervectors, the pointwise multiplication can be realized by an exclusive or (XOR) gate.



(a) **A**+**B**+**C**=**X**. (b) **A**+**B**=**X** in favor of 0. (c) **A**+**B**=**X** in favor of 1.

Figure 2.2: Hamming distance distribution of addition for 10,000-bit hypervectors over 3000 cases. (a) Addition over an odd number of hypervectors; (b) and (c) shows the addition over even number favoring 0 and 1, respectively.

Addition

Pointwise addition, also referred to as *bundling*, computes a hypervector $\mathbf{Z}$ using Eq. (2.4) from the input hypervectors $\{\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_n\}$. Compared to random hypervectors, the generated $\mathbf{Z}$ is maximally similar to the n inputs $\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_n$, i.e., Hamming distance between $\mathbf{Z}$ and any of the n inputs is at a minimum.

$$\mathbf{Z} = [\mathbf{X}_1 + \mathbf{X}_2 + \cdots + \mathbf{X}_n], \tag{2.4}$$

where $[\cdot]$ indicates the sum hypervector $\mathbf{Z}$ is thresholded and binarized to $\{0,1\}^d$ based on the *majority rule*. For convenience, Eq. (2.5) shows an example for the pointwise addition of three 10-bit binary vectors.

$$\begin{aligned} \mathbf{A} &= 0\,0\,0\,0\,1\,1\,0\,0\,1\,1, \\ \mathbf{B} &= 1\,0\,1\,1\,0\,0\,0\,1\,0\,1, \\ \mathbf{C} &= 0\,0\,1\,0\,1\,0\,1\,1\,0\,1, \\ \hline [\mathbf{A}+\mathbf{B}+\mathbf{C}] &= 0\,0\,1\,0\,1\,0\,0\,1\,0\,1. \end{aligned} \tag{2.5}$$

Generally speaking, the addition over odd number of hypervectors has no ambiguity, whereas the addition over an even number can favor either 0 or 1 using the majority function defined in Eq. (2.6). However, this approach may lead to a biased result for adding two hypervectors. Therefore, the bias in adding even number of hypervectors is usually reduced by adding an extra random vector [40]. Figure 2.2 illustrates addition of 10,000-dimensional random hypervectors repeated for 3,000 times. Comparing Figure 2.2(b) to Figure 2.2(c), we see that specifying in favor of 0 or 1 has little impact over addition. It can be observed from Figure 2.2 that the sum is nearly equally similar to the input operands.

$$\text{Majority}(p_1, \cdots, p_n) = \begin{cases} \lfloor \frac{1}{2} + \frac{(\sum_{i=1}^{n} p_i) - \frac{1}{2}}{n} \rfloor, \text{ favor } 0, \\ \lfloor \frac{1}{2} + \frac{\sum_{i=1}^{n} p_i}{n} \rfloor, \text{ favor } 1. \end{cases} \tag{2.6}$$

Multiplication

Pointwise multiplication, also called *binding*, aims to form associations between two related hypervectors. **A** and **B** are bound together to form $\mathbf{X} = \mathbf{A} \oplus \mathbf{B}$, which is approximately orthogonal to both **A** and **B**, where $\oplus$ represents the XOR operation. Eq. (2.7) shows the pointwise multiplication of two 10-bit binary vectors. In a more general case, as shown in Figure 2.3, for two randomly generated 10,000-bit binary hypervectors, their pointwise multiplication result is dissimilar to both of them.

$$\begin{aligned} \mathbf{A} &= 0\,0\,0\,0\,1\,1\,0\,0\,1\,1, \\ \mathbf{B} &= 1\,0\,1\,1\,0\,0\,0\,1\,0\,1, \\ \hline \mathbf{A} \oplus \mathbf{B} &= 1\,0\,1\,1\,1\,1\,0\,1\,1\,0. \end{aligned} \tag{2.7}$$

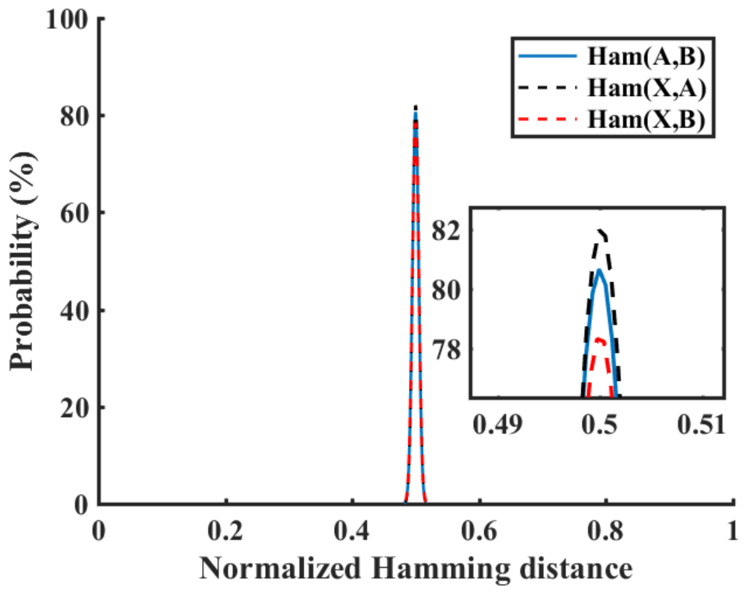


Figure 2.3: Hamming distance distribution of multiplication $\mathbf{X} = \mathbf{A} \oplus \mathbf{B}$ for 10,000-bit hypervectors over 3000 cases.

Permutation

Permutation ρ is a unique unary operation for HDC, which shuffles the hypervector, let us say $\mathbf{A}$. The resulting permuted hypervector $\rho(\mathbf{A})$ is quasi-orthogonal to the initial $\mathbf{A}$, i.e, the normalized Hamming distance is close to 0.5. Mathematically, permutation can be realized by multiplying a permutation matrix. As a specific permutation, a circular shift is widely employed for its friendly hardware implementation. Eq. (2.8) shows a circular shift of a 10-bit binary vector with $\text{Ham}(\mathbf{A}, \rho(\mathbf{A})) = 0.4$. The expected Hamming distance is supposed to be 0.5 for ultra-wide hypervectors. Figure 2.4 indicates the permutation result shows dissimilarity with the original 10,000-bit hypervector.

$$\begin{array}{r} \mathbf{A} = 0\;0\;0\;0\;1\;1\;0\;0\;1\;1, \\ \hline \rho(\mathbf{A}) = 1\;0\;0\;0\;0\;1\;1\;0\;0\;1. \end{array} \tag{2.8}$$

Examples. We illustrate applications of the above operations. For more details, please refer to [15]. Assume that $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{P}, \mathbf{S}, \mathbf{X}, \mathbf{Y}, \mathbf{Z}$ represent 10,000-d random hypervectors:

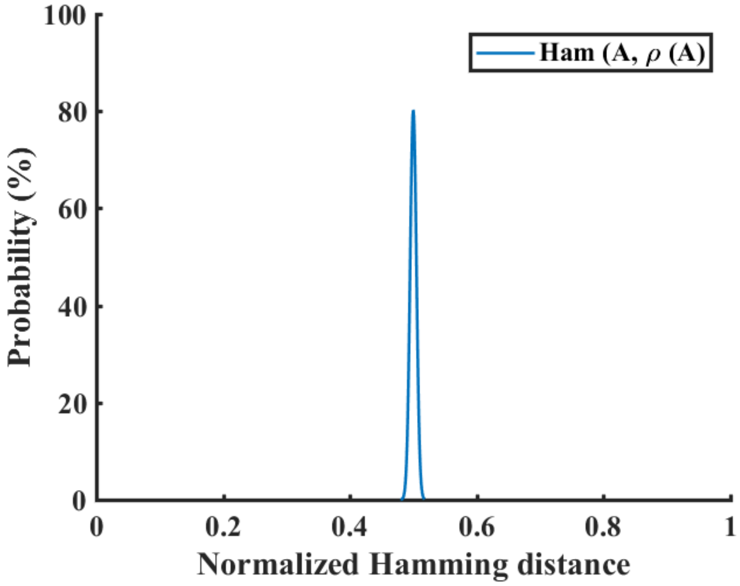


Figure 2.4: Hamming distance distribution of permutation for 10,000-bit hypervectors over 3000 cases.

- *Encode* a pair: To encode "$x = a$", where x is a variable with numerical value same as a, use multiplication to bind their corresponding hypervectors $\mathbf{X}$ and $\mathbf{A}$. The encoding is represented by the generated hypervector $\mathbf{P} = \mathbf{X} \oplus \mathbf{A}$.

- *Release* the value from the pair:

$$\mathbf{X} \oplus \mathbf{P} = \underbrace{\mathbf{X} \oplus (\mathbf{X} \oplus \mathbf{A})}_{\mathbf{X}\oplus\mathbf{X} \text{ cancels out}} = \mathbf{A} \tag{2.9}$$

- *Represent* a set: Given the set $s = \{a, b, c\}$, we have

$$\mathbf{S} = [\mathbf{A} + \mathbf{B} + \mathbf{C}] \tag{2.10}$$

- *Encode* a data record: Given a record with a set of bound pairs $d =$‘$(x = a)\&(y = b)\&(z = c)$’, the record is encoded as:

$$\mathbf{D} = [\mathbf{X} \oplus \mathbf{A} + \mathbf{Y} \oplus \mathbf{B} + \mathbf{Z} \oplus \mathbf{C}] \tag{2.11}$$

- *Extract* the value from a record: To retrieve the value of x:

$$\begin{aligned}\mathbf{A}' &= \mathbf{X} \oplus \mathbf{D} \\ &= \underbrace{\mathbf{X} \oplus [\mathbf{X} \oplus \mathbf{A} + \mathbf{Y} \oplus \mathbf{B} + \mathbf{Z} \oplus \mathbf{C}]}_{\text{distributed}} \\ &= \mathbf{X} \oplus \mathbf{X} \oplus \mathbf{A} + \mathbf{X} \oplus \mathbf{Y} \oplus \mathbf{B} + \mathbf{X} \oplus \mathbf{Z} \oplus \mathbf{C} \\ &= \underbrace{\mathbf{X} \oplus \mathbf{X} \oplus \mathbf{A}}_{=\mathbf{A}} + \underbrace{\big(\mathbf{X} \oplus \mathbf{Y} \oplus \mathbf{B} + \mathbf{X} \oplus \mathbf{Z} \oplus \mathbf{C}\big)}_{\text{noise}} \\ &\approx \mathbf{A}\end{aligned} \tag{2.12}$$

- *Encode* a sequence: Given (a, b), then

$$\mathbf{AB} = \rho(\mathbf{A}) \oplus \mathbf{B} \tag{2.13}$$

- *Extend* the sequence: Extend (a, b) to (a, b, c) using:

$$\begin{aligned}\mathbf{ABC} &= \rho(\mathbf{AB}) \oplus \mathbf{C} \\ &= \rho\big(\rho(\mathbf{A})\big) \oplus \rho(\mathbf{B}) \oplus \mathbf{C}\end{aligned} \tag{2.14}$$

- *Extract* the first element of the sequence:

$$\begin{aligned}&\rho^{-1}\rho^{-1}(\mathbf{ABC} \oplus \mathbf{BC}) \\ &= \rho^{-1}\rho^{-1}(\rho\big(\rho(A)\big) \oplus \rho(\mathbf{B}) \oplus \mathbf{C} \oplus \rho(\mathbf{B}) \oplus \mathbf{C}) \\ &= \rho^{-1}\rho^{-1}(\rho\big(\rho(\mathbf{A})\big) \\ &= \mathbf{A}\end{aligned} \tag{2.15}$$

  where ρ^{-1} is the inverse operation of permutation ρ.

2.3 Learning and Classification By HDC

The first wave of using HDC for classification started in the 1990s [41–44]. The current applications of HDC for classification can be interpreted as the second wave.

2.3.1 The HD Classification Methodology

A system diagram for the classification tasks using HDC is shown in Figure 2.5. In general, (a) during the learning phase, the encoder employs randomly generated hypervectors (pre-stored in the item memory) to map the training data into HD space. A total of k class hypervectors are trained and stored in the associative memory. (b) During the inference phase, the encoder generates the query hypervector for each test data. Then the similarity check is conducted in the associative memory between the query hypervector and every pre-trained class hypervector. Finally, the label with the highest similarity is returned.

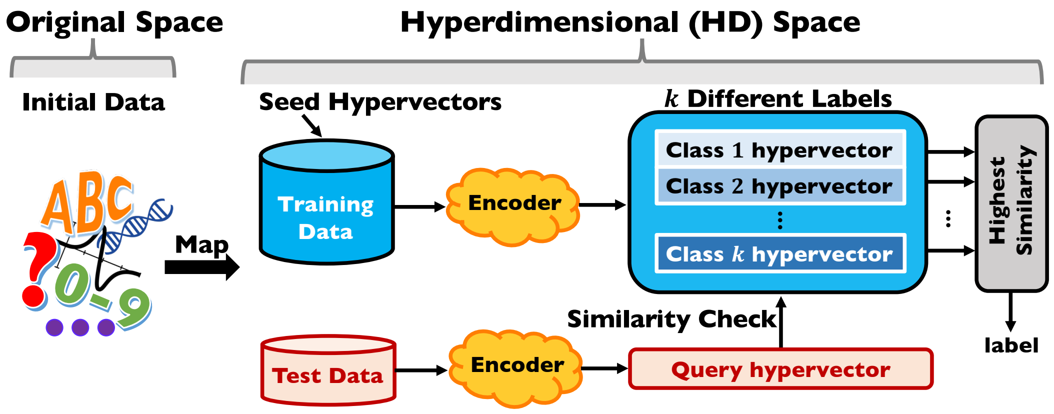


Figure 2.5: Classification overview with HDC [5].

2.3.2 Encoding Methods for HDC

HDC can address various types of input data, including letters, signals, and images. However, we need to map those input data into hypervectors, and this process corresponds to encoding. The encoding process is somewhat similar to the extraction of features. Among the existing HD algorithms, the two encoding methods commonly used include *record-based encoding* and N*-gram-based encoding*. A toy example related to speech signals is used for illustration.

Using Mel-frequency cepstral coefficients (MFCCs) [45], the voice information stored in continuous signals can be mapped into the frequency domain. A feature vector with N elements can be obtained. Each element has its feature value, which is evenly discretized or quantized from $\{F_{min}, F_{max}\}$ to m different levels.

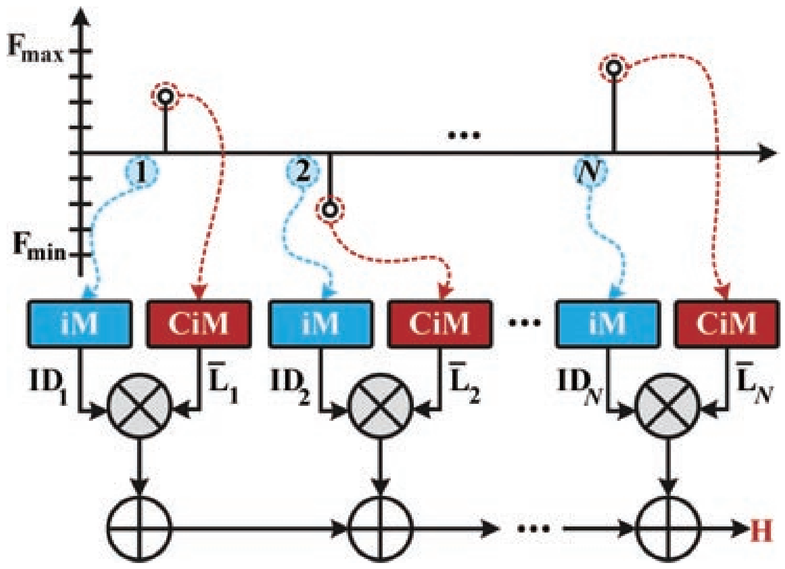


Figure 2.6: Record-based encoding [6]. Note iM refers to item memory, which stores the position hypervectors, and CiM refers to continuous item memory [3], which stores level hypervectors.

Record-based Encoding

This encoding method employs two types of hypervectors, representing the feature position and feature value, respectively. It may be noted that a variation of record-based encoding based on permutations and a chain of binding operations was proposed in [46]. In this encoding, **position hypervectors** ID_i are randomly generated to encode the feature position information in a feature vector, where $1 \leq i \leq N$. The feature value information is quantized to m **level hypervectors** $\{\mathbf{L}_1, \mathbf{L}_2, \cdots, \mathbf{L}_m\}$. For an N-dimensional feature, a total of N level hypervectors $\bar{\mathbf{L}}_i$ should be generated, which are chosen from m level hypervectors $\{\mathbf{L}_1, \mathbf{L}_2, \cdots, \mathbf{L}_m\}$ based on the feature value. Note that, position hypervectors ID_i are orthogonal to each other, while level hypervectors $\{\mathbf{L}_1, \mathbf{L}_2, \cdots, \mathbf{L}_m\}$ are supposed to have correlations between the neighbours. To realize this, in [47] the first level hypervector $\mathbf{L}_1$ represents the feature value F_{min}. Then each time, d/m randomly selected bits are flipped to generate the next level hypervector, where d is the dimensionality of the hypervectors. The continuous bit-flipping was first introduced in [6] and later followed by other use cases [8, 48, 49]. This bit-flipping approach ensures the correlations between neighbor levels, while the last level hypervector $\mathbf{L}_m$ is nearly orthogonal to $\mathbf{L}_1$. The encoding occurs by binding each position hypervector with its level hypervector. As described in Eq. (2.16), the final encoding hypervector $\mathbf{H}$ can be obtained by adding these results together. The entire encoding process is illustrated in Figure 2.6.

$$\begin{aligned} &\mathbf{H} = \bar{\mathbf{L}}_1 \oplus \mathrm{ID}_1 + \bar{\mathbf{L}}_2 \oplus \mathrm{ID}_2 + \cdots + \bar{\mathbf{L}}_N \oplus \mathrm{ID}_N, \\ &\bar{\mathbf{L}}_i \in \{\mathbf{L}_1, \mathbf{L}_2, \cdots, \mathbf{L}_m\}, \text{ where } 1 \le i \le N. \end{aligned} \tag{2.16}$$

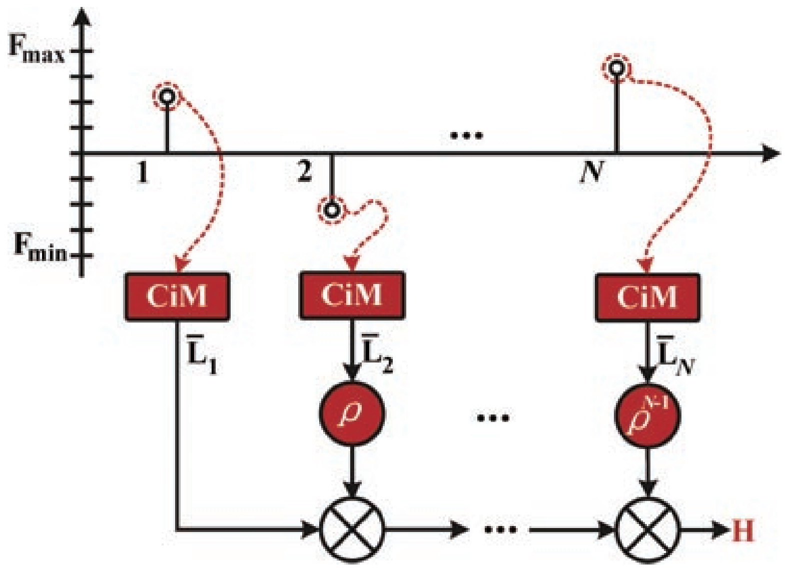


Figure 2.7: N-gram-based encoding [7]. CiM stores level hypervectors that are mutually orthogonal.

N-gram-based Encoding

The method of mapping N-gram statistics into hypervectors was proposed in [17]. First random level hypervectors are generated. Then the feature values are obtained by permuting these level hypervectors in this encoding method. For example, the level hypervector $\bar{\mathbf{L}}_i$ corresponding to the i-th feature position is rotationally permuted by $(i-1)$ positions, where $1 \le i \le N$. We can get the final encoded hypervector $\mathbf{H}$ by Eq.

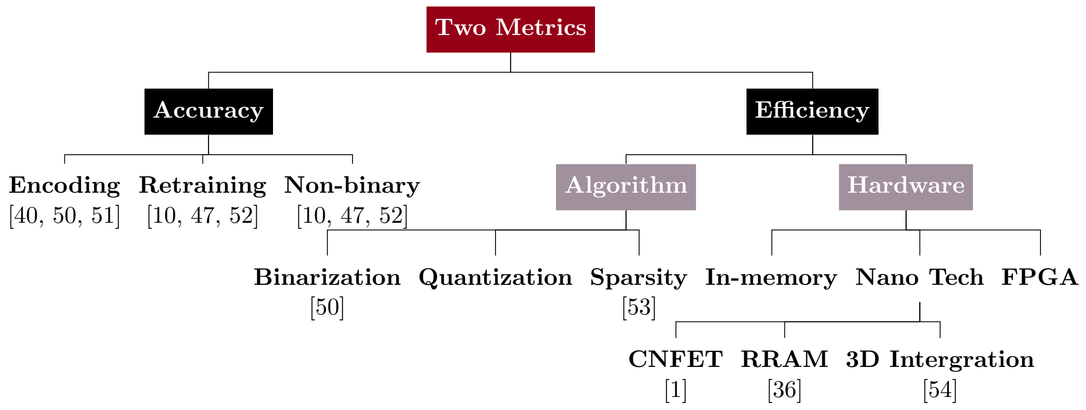


Figure 2.8: Two benchmarking metrics in HDC and some possible ways to improve these metrics.

(2.17). Such an encoding process is illustrated in Figure 2.7.

$$\begin{aligned} &\mathbf{H} = \bar{\mathbf{L}}_1 \oplus \rho\bar{\mathbf{L}}_2 \oplus \cdots \oplus \rho^{N-1}\bar{\mathbf{L}}_N, \\ &\bar{\mathbf{L}}_i \in \{\mathbf{L}_1, \mathbf{L}_2, \cdots, \mathbf{L}_m\}, \text{ where } 1 \le i \le N. \end{aligned} \tag{2.17}$$

As stated in [7], for speech recognition, the N-gram-based encoding method achieves lower accuracy than record-based counterpart. This encoding method is also used to address data types of letters, such as language recognition [5] and DNA sequencing [18].

2.3.3 Benchmarking Metrics in HDC

In HDC, there is always a tradeoff between accuracy and efficiency, e.g., see [50]. As shown in Figure 2.8, a large amount of work has been carried out to improve the classification accuracy, energy efficiency, or both at the same time.

Accuracy

In terms of accuracy, the encoding method plays a significant role since each encoding may not be efficient for different types of data. Good encoding for HD to achieve high accuracy is hard [55]. In this sense, an appropriate choice of encoding method can improve the accuracy. Efficient encoding approaches have been presented in [56]. The approach in [7] integrates different encoding methods together to achieve higher accuracy at the expense of hardware area. Compared to single-pass training, retraining iteratively improves the training accuracy [8]. Thus the classification accuracy is improved by using a more accurately trained model. Moreover, using binary hypervectors may degrade the accuracy. Hence with enough resources, non-binary models can be used to achieve high accuracy.

Efficiency

For efficiency, improvements mainly focus on algorithm and hardware characteristics. From the algorithm perspective, dimension reduction is the most natural way to realize efficiency. Simulations show that by slightly reducing the dimensionality of hypervectors, the classification accuracy still remains in an acceptable range but saves hardware resources [47]. Binarization, which refers to employing binary hypervectors instead of non-binary model, accelerates computation and reduces hardware resources [51]. The precision is degraded by quantizing the non-binary HD model. QuantHD has been proposed in [47] to achieve higher efficiency with minimal impact on accuracy. Sparsity was introduced in HDC in the framework of BSDC [57]. Tradeoff between dense and sparse binary vectors has been presented in [50]. By introducing the concept of sparsity to hypervector representation, [53] proposes a novel platform, SparseHD, which reduces inference computations and leads to high efficiency. From the hardware perspective, HDC involves a large number of bit-wise operations, as well as the same computation flow for different HD applications, making FPGA a nice platform for hardware acceleration [58]. Moreover, as proposed in [59], combining HDC with the concept of in-memory computing, which is featured as RAM storage and parallel distribution, may create opportunities for HD acceleration. Additionally, several emerging nanotechnologies, including carbon nanotube field-effect transistors (CNFETs) [1], resistive RAM (RRAM) [36], and monolithic 3D integration [54], have demonstrated implementations of HDC at high speed [1]. Dimensionality reduction has been evaluated in an actual prototyped system using vertical RRAM (VRRAM) in-memory kernels in [60].

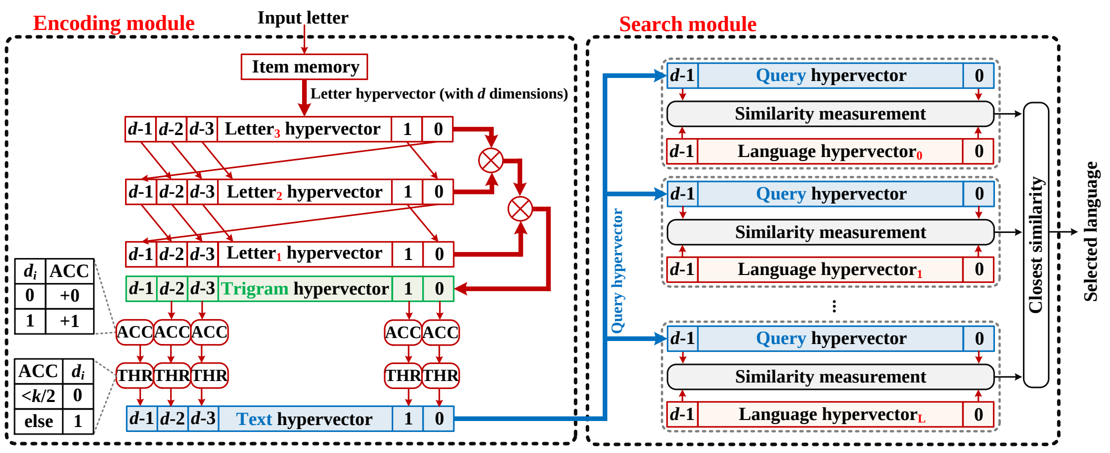


Figure 2.9: The architecture for language recognition with HDC [1, 5].

2.4 Applications in HD Classification

In what follows, some classical HDC applications in classification tasks as well as several novel design approaches that can balance tradeoff of accuracy and efficiency are described. They are categorized based on their input data types, namely letters, signals and images.

2.4.1 Letters

European Language Recognition Using HDC

HDC for European language recognition was first explored by [17]. Literature [1] presents an HDC nanosystem, which implements HD operations based on emerging nanotechnologies—CNFETs, RRAM and 3D integration—offering large arrays of memory and resulting in reduction of energy consumption. From its three-letter sequence called *trigrams*, such a nanosystem can identify the language of a given sentence [1]. Define a profile by a histogram of trigram frequencies in the unclassified text. The basic idea is to compare the trigram profile of a test sentence with the trigram profiles of 21 languages, and then find the target language which has the most similar trigram profile [17].

- **Baseline.** Scan through the text and count the trigram to compute a profile.

A total of $27^3 = 19,683$ trigrams are possible for the 26 letters and the space. Thus the trigram counts can be encoded into a 19,683-dimensional vector and such vectors can be compared to find the language with the most similar profile. However, this straightforward and simple approach generalizes poorly. Specifically, compared to trigrams, higher-order N-grams will have higher complexity. For example, the number of possible pentagrams is $27^5 = 14,348,907$.

- **HD classification algorithm.** (a) Choose a set of 27 letter hypervectors randomly, serving as the seed hypervector. Note that all training and test data employ the same seeds. In this design, the dimensionality is selected to be 10,000. (b) Generate trigram hypervectors with permutation and multiplication. For example, let (a, b, c) represent a trigram. Then rotate the hypervector A twice, hypervector B once, and use hypervector C with no change, and then multiply them component by component as described in Eq. (2.14). (c) The target profile hypervector is then the sum of all the trigram hypervectors in the text. (d) Compare the profile of a test sentence to the language profiles, and return the most similar one as the classification result.

Compared to the baseline algorithm, the HD algorithm generalizes better to any N-gram size when 10,000-dimensional hypervectors are used.

The HD classification hardware architecture for language recognition using trigrams proposed in [5] is shown in Figure 2.9. Two main modules are implemented. They include the encoding module and the search module. (a) The encoding module takes a stream of letters as the input. Each letter is mapped to the HD space and its corresponding randomly generated hypervector is stored in the item memory. Here it addresses the trigrams where each group of three hypervectors produces a trigram hypervector. Accumulate those trigram hypervectors and perform the majority operation using the threshold to generate a text hypervector. (b) During the training phase, a total of 21 text hypervectors are trained as the learned class hypervectors and are stored in the associative memory in the search module. During the testing phase, the encoding module generates the text hypervector as a query hypervector. This query hypervector is then broadcast to the search module and compared to the stored class hypervectors to predict the language label, which has the closest similarity. As listed in Table 2.4, the

HD classifier achieves 96.70% accuracy.

Using the same architecture shown in Figure 2.9, and combining with the emerging nanotechnologies—CNFETs, RRAM and their monolithic 3D integration—the HDC hardware implementation achieves classification accuracy up to 98% for over $> 20,000$ sentences [1].

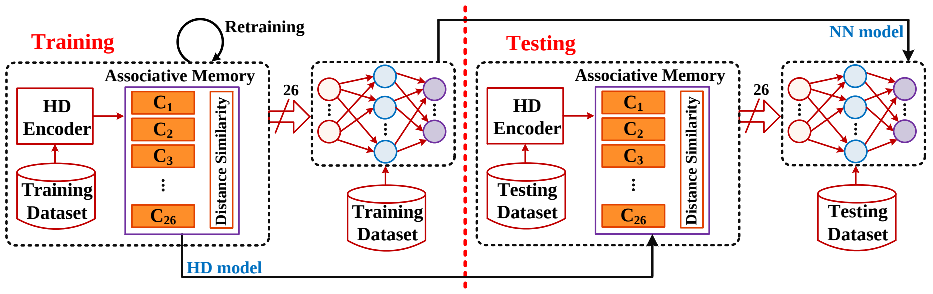


Figure 2.10: VoiceHD+NN flow for training and testing [8].

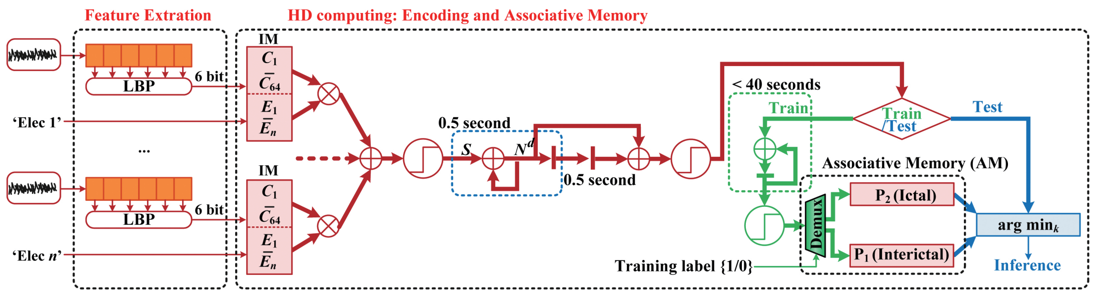


Figure 2.11: The architecture for Laelaps with HDC to detect and alarm seizure [9].

2.4.2 Signals

HDC Classification for Speech Recognition

The development of the Internet of Things (IoT) has motivated the market need for speech recognition. Though deep neural networks (DNNs) have been widely used for speech recognition, it requires expensive hardware and high energy consumption. This has inspired research for speech recognition based on HDC which can achieve fast computation and energy efficiency.

In [8], VoiceHD, a new speech recognition technique, is proposed for classifying 26 letters from the spoken dataset. At the beginning, the voice signal is transformed to the frequency domain, which contains N frequency ID channels and M levels. Then VoiceHD maps these ID and level information into random hypervectors stored in the item memory. Combining these hypervectors, in the training phase, VoiceHD encoding module generates the learned patterns corresponding to 26 hypervectors that are stored in the associative memory. In the testing phase, VoiceHD uses the same encoding module to generate the query hypervector, which is broadcast to the associative memory. Comparing the query hypervector with the stored 26 class hypervectors, the hypervector with maximum similarity is retrieved to predict the letter. Here, dimensionality d of the hypervectors is $10,000$.

Researchers tested their VoiceHD design over Isolet dataset [61], where a total of 150 subjects spoke the name of each letter of the alphabet twice. The key findings are as follows: (a) Varying the value of M, the number of levels of the amplitude between -1 and 1, with N, the number of frequency bins, fixed at 617, the recognition accuracy increases with increase in M. Note the encoding efficiency degrades with large $M > 10$. The maximum accuracy reaches 88.4% using $M = 10$. (b) To improve the classification accuracy, researchers retrain the associative memory by modifying the trained class hypervectors. The accuracy can be improved to 93.8%. (c) Combining VoiceHD with a small neural network, the corresponding VoiceHD + NN flow is shown in Figure 2.10. Such a small NN has three layers. There are 26 neurons in the first layer, 50 neurons in the hidden layer and another 26 neurons in the last layer. The classification accuracy can be improved to be 95.3%. (d) Compared to the pure NN with 93.6% classification accuracy, VoiceHD, and VoiceHD+NN show 4.6× and 2.9× faster training speed, 5.3× and 4.0× faster testing speed, and 11.9× and 8.6× higher energy efficiency, respectively.

Seizure Detection Using HDC

The Laelap algorithm, which utilizes local binary pattern (LBP) codes to conduct the feature extraction from iEEG signals, has been proposed in [9] for seizure prediction. Here HDC is applied to capture the statistics of the time-varying LBP codes for all the electrodes. Figure 2.11 illustrates the complete processing chain. (a) Since the down-sampling frequency is 512 Hz, thus every one second (1s) data contains 512 samples. Among these samples, the sampled iEEG signals are encoded to 6-bit LBP codes. This completes the feature extraction part. (b) It utilizes record-based encoding, where two types of hypervectors are randomly generated. Specifically, each LBP code is transformed to a d-dimensional hypervector $\mathbf{C}_i$, while the hypervectors $\mathbf{E}_i$ are used to represent the corresponding electrode name. For every new sample, the hypervectors $\mathbf{E}_i$ and $\mathbf{C}_i$ are bound together to form a composite hypervector $\mathbf{S} = [\mathbf{C}_1 \oplus \mathbf{E}_1 + \cdots \mathbf{C}_n \oplus \mathbf{E}_n]$, where n is the number of electrodes for a specific patient. Then the histogram of LBP codes $\mathbf{H}$ is computed for a moving window of 1s with 0.5s overlap. Therefore the composite hypervector $\mathbf{H} = [\mathbf{S}^1 + \mathbf{S}^2 + \cdots + \mathbf{S}^{512}]$ is updated every 0.5s. (c) For learning, two prototype hypervectors $\mathbf{P}_1$ and $\mathbf{P}_2$ should be trained. For the interictal prototype vector $\mathbf{P}_1$, all $\mathbf{H}$ computed over the 30s should be accumulated and normalized to be stored in the associative memory. Depending on the seizure's duration, the ictal prototype vector $\mathbf{P}_2$ is generated using all $\mathbf{H}$ over an ictal state, which may last 10s to 30s. (d) For classification, comparing $\mathbf{P}_k$ with a query $\mathbf{H}$, the label is updated every 0.5s with the shortest Hamming distance $\text{Ham}(\mathbf{H}, \mathbf{P}_k)$, where $k = 1, 2$. (e) The algorithm also generates the seizure alarm. In postprocessing, if the last 10 labels all indicate $\mathbf{P}_2$ ($t_c = 10$) and the distance score $\Delta > t_r$, then the seizure alarm is generated.

The evaluation shows the Laelaps algorithm outperforms other machine learning methods, such as SVM, in terms of energy efficiency. It is worth noting that many simpler seizure detection and prediction algorithms have been proposed in the literature [62–66]. A fair comparison of classifier accuracy between HD and traditional classification needs to be explored in the future.

Quantization in HDC

In dealing with signals, HDC usually makes use of floating point models to improve the classification accuracy at the cost of high computation cost. In [47], QuantHD is proposed as a quantization of HD model, which projects the trained non-binary hypervectors to a binary or ternary model, with elements in $\{0, 1\}$ or $\{-1, 0, +1\}$, to represent class hypervectors. To compensate the accuracy degradation caused by quantization, a retraining approach is used where an iteration number of 30 is pre-defined. The similarity check is no longer cosine metric (non-binary model), but Hamming distance (binary model) or dot product (ternary model). Compared to the existing binarized HDC, such QuantHD improves on average 17.2% accuracy with a similar computation cost.

HDC Using Model Compression

As a mathematical framework, HDC can be an alternative for machine learning problems. This was envisioned in [67]. Due to the high dimensionality, the inference of HDC is quite expensive, especially when it is applied to embedded devices with limited resources. For example, the memory is limited. Therefore, reducing the high dimensionality of hypervectors without sacrificing the accuracy has been investigated in [10]. Thus, CompHD is a general method that compresses the model size with minimal loss of accuracy. The addressed hypervectors are in $\{-1, 1\}^d$. Instead of Hamming distance, the similarity metric in CompHD is cosine similarity.

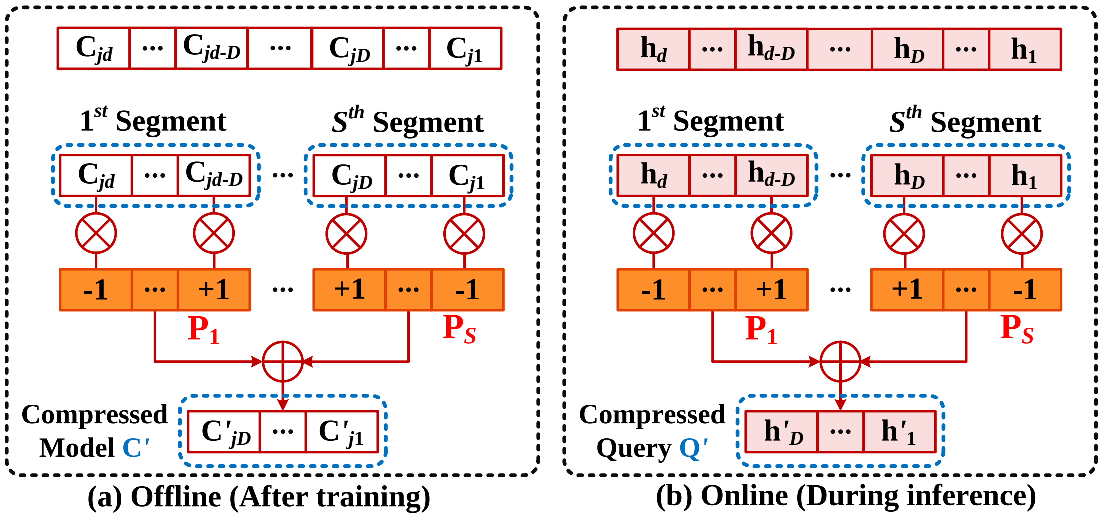


Figure 2.12: CompHD for (a) an HD model and (b) a query data [10].

To reduce the HD model size, it is natural to use low-dimensional hypervectors. However, experimental results of three practical applications using different dimensionalities in HD classification show that the efficiency is improved by reducing model size at the cost of accuracy.

To maintain high accuracy when reducing the dimensionality, the proposed CompHD employs the architecture shown in Figure 2.12. With no reduction in model size, $\mathbf{C}_i$ represents the class hypervector, $\mathbf{Q}$ represents the query hypervector, where $1 \leq i \leq k$. In CompHD, class hypervectors, and query hypervectors are compressed, which means the original hypervectors are divided into s segments. To store most of the information in original hypervectors with the full size, using Hadamard method [68], CompHD generates $\mathbf{P}_1, \mathbf{P}_2, \cdots, \mathbf{P}_s$, which are in $\{-1, 1\}^D$ and are orthogonal to each other, where $D = d/s$. Specifically, the compressed class hypervector $\mathbf{C}'$ and query hypervector $\mathbf{Q}'$ are calculated using multiplication and addition in HD as described by Eq. (2.18). By doing so, only little information is lost when we compress the model size and high accuracy can be maintained.

$$\mathbf{C}' = \sum_{i=1}^{s} \mathbf{P}_i \mathbf{C}^i, \qquad \mathbf{Q}' = \sum_{i=1}^{s} \mathbf{P}_i \mathbf{Q}^i \tag{2.18}$$

Their evaluation shows that, compared to the original HD classification that purely reduces the dimensionality with the compression factor $s = 20$, the classification accuracy for the three applications is still in an acceptable range. In particular, maintaining the same accuracy as the original, CompHD can on average reduce the model size by 69.7% while still achieving 74% energy improvement and 4.1× execution time speedup in the context of activity recognition, gesture recognition, and valve monitoring applications [10]. Therefore, CompHD is suitable for low-power IoT devices to achieve higher efficiency with comparable accuracy.

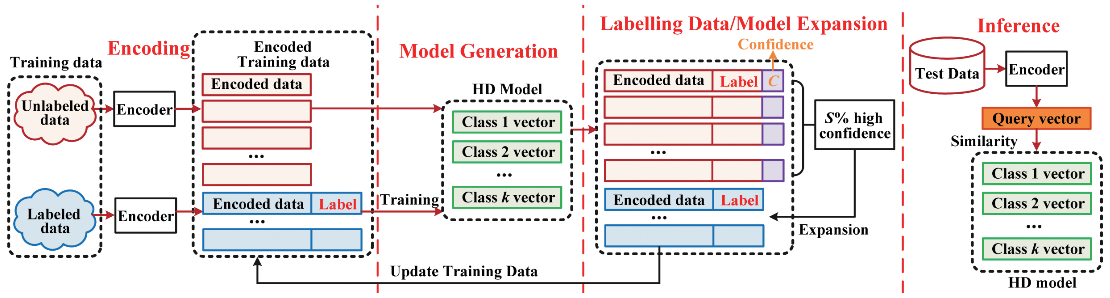


Figure 2.13: Overview of SemiHD framework supporting self-training in HD space [11].

Adaptive Efficient Training for HDC

Single-pass training leads to low accuracy. To improve this, iterative training might be one efficient solution. However, a lack of controllability of training iterations in HD classification may result in slow training or divergence. To solve this training issue, [52] proposes a retraining approach, AdaptHD.

The basic idea is illustrated as follows: (a) Conduct the initial training by using binary hypervectors to generate the non-binary class hypervectors. (b) Retrain the class hypervectors by looking at the similarity of each trained class hypervector ($\mathbf{C}$) with the training hypervector ($\mathbf{H}$). Update the model using Eq. (2.19) if the current training hypervector leads to a misclassification error. Otherwise, there is no change. For example, there is a mismatch if $\mathbf{H}_i$ is supposed to belong to $\mathbf{C}_{\text{correct}}$ but is classified as $\mathbf{C}_{\text{wrong}}$, where $\mathbf{C}_{\text{correct}}$ and $\mathbf{C}_{\text{wrong}}$ denote different class hypervectors and $\mathbf{H}_i$ represents the ith training hypervector. (c) After convergence, which means the last three iterations of retraining show less than 0.1% accuracy change, then binarize the final trained model for inference.

$$\begin{cases} \mathbf{C}_{\text{wrong}} = \mathbf{C}_{\text{wrong}} - \alpha \mathbf{H}_i, \\ \mathbf{C}_{\text{correct}} = \mathbf{C}_{\text{correct}} + \alpha \mathbf{H}_i. \end{cases} \tag{2.19}$$

Insights are gained by their results: (a) Small α needs more iterations to get the near-best accuracy. The smooth curve indicates small α is better for fine-tuning. (b) Large α gets to the near-best accuracy much faster, but its high fluctuation may lead to divergence. Based on these two findings, AdaptHD uses large α first to get the near best accuracy faster, then changes to smaller α for fine-tuning until convergence. This is similar to adjusting the step size in the normalized least mean square (LMS) algorithm [69]. AdaptHD offers three types of adaptive methods:

- Iteration-dependent AdaptHD. The change of value α depends on iterations. In the beginning, α starts with a large α_{max}. The learning rate α changes based on the average error rate in the previous β iterations. If the error rate decreases, indicating convergence, then use smaller α; otherwise, increase α.

- Data-dependent AdaptHD. The value α differs in a certain iteration for all data points, and it changes depending on the similarity of the data point with the class hypervectors. Large distance uses large α to reduce the difference.

- Hybrid AdaptHD. Combining the two models, hybrid AdaptHD can achieve high accuracy as iteration-dependent AdaptHD and fast speedup as the data-dependent AdaptHD.

The evaluation shows that, compared to the existing HD algorithm, the hybrid

AdaptHD can achieve 6.9× speedup and 6.3× energy-efficiency improvement.

A Binary Framework for HDC

Generally speaking, HD classification using binary hypervectors shows lower accuracy but higher energy efficiency than non-binary ones. This is because the non-binary framework makes use of the costly cosine similarity rather than the hardware-friendly Hamming distance metric. In [51], BinHD uses three main blocks, encoding, associative search, and counter modules, dealing with binary hypervectors. Their evaluation shows that, over four practical applications, the proposed BinHD can reach 12.4× and 6.3× energy efficiency and speedup in the training process, while 13.8× and 9.9× during the inference, compared to the state-of-art HDC algorithm with comparable classification accuracy.

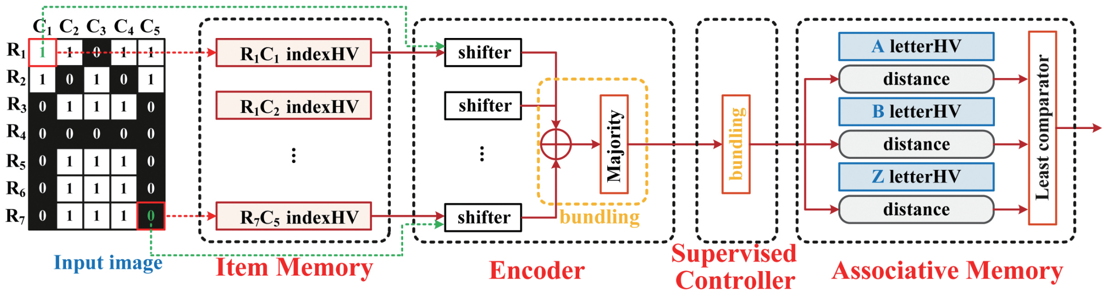


Figure 2.14: Block diagram of the HD Character Recognition System [12].

HDC for Semi-Supervised Learning

In [11], SemiHD has been proposed as a self-training or self-learning approach for semi-supervised learning, where the training data is composed of a small portion of labeled data and a large portion of unlabeled data.

The SemiHD framework is depicted in Figure 2.13 and the flow is illustrated as follows. (a) Encode all the data points, labeled and unlabeled, into HD space with $d = 10,000$ dimensions. (b) Start training from the labeled data to generate k hypervectors, each representing one class. (c) Predict the label for unlabeled data points. Labeling is performed by checking the similarity of unlabeled data with all the class hypervectors, and return the label which shows the highest similarity. (d) Select and add S% of unlabeled data with highest confidence to labeled data, where S is defined as the expansion rate. In [11], typically $S = 5$. (e) Redo the training task based on the expanded labeled data. Such iterative process stops when the accuracy does not change more than 0.1%. (f) Once the model has already been trained, perform the inference task by comparing the similarity of each test data with the trained model, to return the label with maximum similarity.

Their evaluation shows that the SemiHD can on average improve the classification of supervised HD by 10.2%. Additionally, compared to the best CPU implementation, the FPGA counterpart of SemiHD offers 7.11× faster speed and 12.6× energy efficiency.

HDC for Unsupervised Learning

HDC has also been used in several unsupervised applications. See [70–74].

2.4.3 Images

HD Classification for Character Recognition

HD classification has been used for character recognition in [75] and later in [12]. As shown in Figure 2.14, the input image is composed of $7 \times 5 = 35$ pixels. Each pixel has two possible values, that is 0 or 1, representing black or white. (a) Encode each pixel to a binary hypervector (indexHV). Totally 35 orthogonal indexHVs are stored in the item memory. (b) Based on HoloGN encoding—an encoding method proposed in [75] to address image data using HDC—the indexHV is shifted depending on the pixel value. Accumulate all 35 indexHVs and perform a majority rule by a thresholding block to generate a holoHV for one input image. (c) The supervised controller will only be activated when this HD system conducts supervised learning. Otherwise, the system conducts the one-shot learning. The supervised controller accumulates the holoHVs for the same class and employs the thresholding block and generates the letterHV to be stored in the associative memory. The total number of letterHVs is 26. (d) During the test phase, the query hypervector is generated following the same module with test data. Then the similarity of each query hypervector is computed for all trained letterHVs to find the most similar class.

Results in [12] show that HDC performs well for character recognition. Further

Table 2.3: Summary of the strategies used in HDC for accuracy and efficiency improvement.

Applications	Encode[1]	Model Type[2]			Platform[3]	Accuracy[4]	Acceleration[5]	Motivation	Application
		base/ level	train	test					
QuantHD [47]	1	B/B	B/T	B/T	F, C, G	Re, shuffle	Q, DR, F	speedup+ accuracy	speech, activity, face, phone position
VoiceHD [8]	1	B/B	B	B	C	NN, Re	DR, B	Replace deep learning	speech
CompHD [10]	3	P	N	N	F	N	DR, Comp	DR without accuracy loss	activity, gesture, valve monitoring
AdaptHD [52]	1	B/B	N	B	C	Re, N	B, Adapt	accuracy+ short time Re	speech, face, activity, Cardiotocograms
BinHD [51]	1	B/B	B	B	C	Re	B	speedup	speech, face, activity, Cardiotocograms
SemiHD [11]	1	B/B	B	B	F, C	Re, N	DR, B	Replace deep learning	17 popular datasets [76]
Language [5]	2	B	B	B	F	\	B	energy saving + robustness	language recognition
Character [12]	3	B	B	B	\	Binary	DR	data classification in IoT	character recognition
Laelap [9]	1	B	B	B	C, G	\	\	energy efficiency	seizure detection

[1] three encoding methods. 1: record-based encoding, 2: N-gram-based encoding, 3: a novel method.
[2] symbol "/" is used in record-based encoding. B: binary, P: bipolar, T: Ternary, N: non-binary.
[3] implementation platforms. F: FPGA, C: CPU, G: GPU.
[4] strategies for accuracy improvement. Re: retraining, N: non-binary model, NN: neural network.
[5] strategies for efficiency improvement. DR: dimension reduction, Q: quantization, B: binarization, F: FPGA, Comp: compression, Adapt: adaptive.

optimization for HDC may be conducted by reducing the dimensionality and increasing the input image size. The results also show that HDC offers great robustness against noise. The system of 4,000-bit hypervectors achieves comparable average accuracy to its 12,000-bit counterpart at 0% distortion, and achieves an average accuracy of 89.94% with 14.29% distortion.

2.4.4 Summary

As mentioned above, HDC shows great potential in dealing with data in the form of signals [8, 9, 16, 78, 87], letters [17, 77], and images [12, 36, 75], as long as these can be transformed into the HD space. Such pre-processing may include feature extraction and encoding. The evaluation shows that HDC achieves good results for seizure detection [9,

Table 2.4: Partial list of applications based on HDC[1] in [1].

Applications	**Inputs(#)**[2]	**Classes(#)**[3]	**HDC**	**Baseline**
Language recognition [5, 17]	1	21	96.70%	97.90%
Text categorization [77]	1	8	94.20%	86.40%
Speech recognition [8]	1	26	95.30%	93.60%
EMG gesture recognition [6]	4	5	97.80%	89.70%
Flexible EMG gesture recognition [49]	64	5	96.60%	88.90%
EEG brain-machine interface [78]	64	2	74.50%	69.50%
ECoG seizure detection [16]	100	2	95.40%	94.30%
DNA sequencing [18]	1	\	99.74%	94.53%
Character recognition [12]	1	10	89.94%	\

[1] Other works, like [79–86], are not listed in this table.
[2] represents the number of input data.
[3] represents the number of class hypervectors to be trained and stored in the associative memory.

16]. In addition, HDC can also be combined with quantization technique to binarize HD model with minimal accuracy loss [88]. Table 2.3 offers more details about improvement strategies adopted in HDC for accuracy and efficiency. As can be seen from Table 2.4, HDC offers an acceptable accuracy, but with quite high efficiency. In some applications like DNA sequencing [18], HDC outperforms other machine learning methods.

There still exist some interesting papers not discussed in detail in this review work. Interested readers can refer to the following references, which include but are not limited to: (a) Considering the security issue when IoT devices release the offload computation to the cloud, [89] illustrates how the proposed SecureHD accelerates efficiency with high security. (b) To balance the tradeoff between efficiency and accuracy, QubitHD [88] is proposed as a stochastic binarization algorithm to achieve comparable accuracy to the non-binarized counterparts. SparseHD [53] takes advantage of the sparsity of the trained HD model for acceleration.

HDC is still in its infancy. Future directions may include but is not limited to:

- More cognitive tasks: Inspired by [50], apart from the engineering aspect of HDC, which is to solve classification tasks, more "cognition" aspects of HDC should be explored. Such tasks include but are not limited to analogical reasoning, semantic generalization and relational representation.

- Feature exaction and encoding method: Since HDC cannot directly address data like signals and images, feature exaction is vital to representation of information. For example, [87] partially deals with this by addressing the problem of mapping data to a high-dimensional space.

- Similarity measurement: Though cosine similarity and Hamming distance are currently widely used, new metrics should be developed that are hardware-friendly and can lead to high accuracy.
- Multiple class hypervectors: Traditional classifiers use multi-dimensional features to train a classifier. Often ranking can be used to select a small number of features out of many features [90]. It is possible that multiple class hypervectors, similar to multiple features in traditional classification, can be generated to represent a class in HD classification. Subsequently, multiple query hypervectors will need to be compared with their corresponding class hypervectors for each class. This is a topic for further research.
- Accuracy improvement: Strategies like retraining should be explored to further improve the accuracy of HDC.
- Hardware acceleration: Rebuilding the specific implementation for HDC to store and manipulate a large number of hypervectors may result in high speed and energy efficiency. Moreover, inspired by [50], which discusses tradeoffs related to the density of hypervectors, a choice between dense and sparse approaches should be accordingly made based on the application scenarios. For example, adopting sparse representation requires lower memory footprints.
- General HDC processor: Inspired by [3], addressing different types of data with only one general processor containing a large word-length ALU is of great interest.
- Hybrid systems: Hybrid systems are partially based on HDC and partially on conventional machine learning. Only a few examples exist so far [91–94]. Further research on this topic can be explored in the future.

2.5 Conclusion

This chapter has summarized the fundamental arithmetic operations for the emerging computing model of HDC that might achieve high robustness, fast learning ability, hardware-friendly implementation, and energy efficiency. Mathematically, HDC can be viewed as an alternative in dealing with machine learning problems. Though in its infancy, HDC shows its potential to be used as a lightweight classifier for applications with limited resources. This model can achieve outstanding classification performance for certain problems like DNA sequencing. Balancing the tradeoff between accuracy and efficiency is an important area of research. Improvements include but are not limited to encoding, retraining, non-binary model, and hardware acceleration. HDC sometimes leads to outstanding classification accuracy, while sometimes achieves acceptable accuracy but high efficiency. Thus, users need to evaluate whether HDC is suitable for their application. Additionally, HDC can be used in applications such as seizure detection, speech recognition, character recognition and language detection. More "cognition" aspects of HDC, including analogical reasoning, relationship representation and analysis, will need to be further developed in the future.

Chapter 3

Applicability of seizure detection using HDC

The work for this chapter has been published in [20, 21]. This chapter investigates the applicability of seizure detection using HDC. Two encoding approaches are studied: LBP and PSD encoding.

3.1 Introduction

Seizure detection [9, 16, 62, 95, 96] and prediction [65, 97–101] from either scalp or intra-cranial electroencephalogram (iEEG) are two separate but related classification problems [102]. The baseline EEG is referred as *interictal* whereas the EEG during seizure occurrence is referred as *ictal*. The EEG 30-60 minutes before seizure is referred as *preictal*. Seizure detection is a binary classification problem that classifies ictal *vs.* interictal whereas seizure prediction classifies preictal *vs.* interictal. An implanted seizure prediction device can trigger deep brain stimulation that can avert the seizure. Seizures can be detected or predicted using features such as wavelet coefficients of the signal or the error signal [62, 103], line length [104], power spectral density (PSD) [66, 100], and ratio of band power [63, 66, 105], and classifiers such as SVM [64]. Recently it has been shown that seizures can also be detected or predicted by convolutional neural networks (CNN) [98, 106, 107]. The reader is referred to [108] for a review of approaches for seizure detection. This work addresses the seizure detection problem.

Seizure detection is of interest in two different contexts. First, this can be used to automatically annotate the EEG so that the neurologist's time to annotate the data can be reduced significantly. Second, a subject can be treated using a fast-acting drug when a seizure detection warning is generated by a wearable device.

Seizure detection using HDC and local binary pattern (LBP) encoding has been addressed in prior work [9, 95, 96, 109]. In [56], it was pointed out that power spectral density (PSD) values can be used in the context of HD; however, this should be avoided. A scalar multiplication (weighting) method is used in [56] to map real-valued features (e.g., PSD features) without any quantization. This method directly multiplies the bipolar feature hypervectors by these real-valued features leading to real-valued hypervectors. Since it requires floating-point operations and storage, [56] suggests avoiding using PSD as much as possible. No experimental results were also presented in [56] for seizure detection using PSD.

This research explores the applicability of binary HDC to *patient-specific* seizure detection using the iEEG data from the Kaggle seizure detection contest [110] using two classes of features: LBP and PSD. This research makes five novel contributions. (a) This work investigates the LBP feature, which has been studied before in the context of HDC in [9, 16], for another seizure dataset to test its generalizability as an efficient feature for seizure detection. (b) The band power and ratio of band power computed from PSD have been investigated in the context of traditional machine learning, but not in the context of HDC until our prior conference paper [20]. In contrast to [56], we quantize the PSD features to q levels and use hypervectors to encode these quantized PSD values. In the context of selected PSD features, unlike [20] where the features selected by classification and regression tree (CART) from a prior work are used, in this chapter, features are selected by Fisher score. In the context of HD using PSD features, three approaches are introduced in [20]; these are referred as: *single classifier long hypervector*, *multiple classifiers*, and *single classifier short hypervector*. (c) Only test accuracy is considered as the performance measurement metric in [20], which is insufficient for seizure detection. In this work, four main metrics are used to measure the performance, namely the test accuracy, sensitivity, specificity and the area under this curve (AUC). (d) In this work, to the best of our knowledge, a *hypervector distance* plot is introduced for the first time to visualize the quality of classification results. This plot illustrates the distance of the query hypervector from one class hypervectors *vs.* that from the other. (e) We show that the length of the hypervectors can be reduced from 10,000 bits to 1,000 bits for both LBP and PSD methods. Furthermore, for two approaches of the PSD method, the length of the hypervectors can be as short as 100 bits. A short hypervector can reduce energy consumption significantly. No prior work has shown that hypervectors as short as 100 bits can be used for classification using HDC.

Using iEEG data from the Kaggle contest, it is shown that HDC classifiers using PSD features can achieve better performance than LBP nearly for all subjects. This observation is new. Thus, identifying discriminating features for HDC is equally important. Unlike in CNNs where the network learns the features implicitly, the HDC classifier performance varies with different features. This work is an extended version of [20]. While LBP was not used in [20], this work describes results using LBP.

3.2 Methodology

This section illustrates two different HDC-based learning strategies: LBP and PSD encoding. In general, the LBP method utilizes the time-domain information, whereas the PSD method employs the features extracted from the frequency domain.

3.2.1 LBP Method

This work employs the LBP method proposed by [109] for a different iEEG dataset to test its generalizability for seizure detection. Based on [109], iEEG signals are encoded as LBP codes, which reflect the time-domain information and are able to distinguish ictal and interictal states. LBP codes can be computed as follows: (a) Consecutive iEEG signal samples are converted into a bit stream depending on the sign of the temporal difference of adjacent samples. If the difference is positive, then the corresponding LBP code is assigned as 1; otherwise it is 0. (b) A length-l (l-bit) LBP code is generated from $(l + 1)$ consecutive iEEG signal samples. As an example, we encode 7 points of a

$$\mathbf{hv_{spatial_i}} = \Big[\sum_{j=1}^{N} \bar{\mathbf{hv}}_{\mathbf{LBP_j}} \oplus \mathbf{Ch_j}\Big], \text{ where } 1 \leq j \leq N. \tag{3.1a}$$

$$\mathbf{hv_{win_k}} = \Big[\sum_{i=1}^{W-l} \mathbf{hv_{spatial_i}}\Big], \text{ where } 1 \leq i \leq W - l. \tag{3.1b}$$

$$\mathbf{hv_{class}} = \big[\mathbf{hv_{win_1}} + \cdots + \mathbf{hv_{win_k}} + \cdots + \mathbf{hv_{win_K}}\big], \text{ where } 1 \leq k \leq K. \tag{3.1c}$$

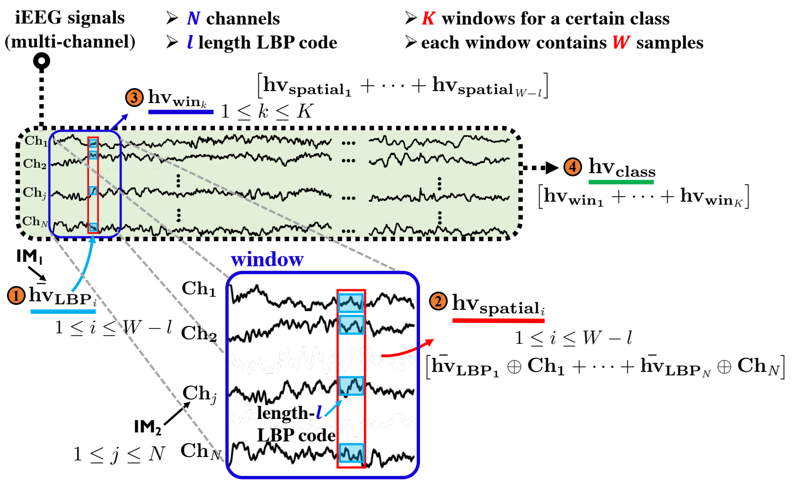

iEEG signals
(multi-channel)
➢ N channels
➢ l length LBP code
➢ K windows for a certain class
➢ each window contains W samples
③ $\mathbf{hv_{win_k}}$
$1 \leq k \leq K$
$[\mathbf{hv_{spatial_1}} + \cdots + \mathbf{hv_{spatial_{W-l}}}]$
④ $\mathbf{hv_{class}}$
$[\mathbf{hv_{win_1}} + \cdots + \mathbf{hv_{win_K}}]$
Ch_1
Ch_2
Ch_j
Ch_N
IM₁
① $\bar{\mathbf{hv}}_{\mathbf{LBP}_i}$
$1 \leq i \leq W - l$
window
② $\mathbf{hv_{spatial_i}}$
$1 \leq i \leq W - l$
$[\bar{\mathbf{hv}}_{\mathbf{LBP_1}} \oplus \mathbf{Ch_1} + \cdots + \bar{\mathbf{hv}}_{\mathbf{LBP}_N} \oplus \mathbf{Ch}_N]$
length-l
LBP code
IM₂
$1 \leq j \leq N$

Figure 3.1: HD classification using LBP method.

time-series raw data as four 3-bit LBP codes as shown in Eq. (3.2).

$$\begin{array}{r}\text{raw data: } 2.0,\ 1.5,\ 1.2,\ 2.3,\ 2.2,\ 3.2,\ 3.5 \\ \text{bit stream: } 0,\ \ 0,\ \ 1,\ \ 0,\ \ 1,\ \ 1 \qquad\qquad \\ \hline \text{LBP code: } (001),\ (010),\ (101),\ (011) \quad\end{array} \tag{3.2}$$

Figure 3.1 shows the HD classification using the LBP method. For a certain class, its N-channel iEEG signals have in total K windows. Each window contains W samples. Before training, hypervectors $\{\mathbf{hv_{LBP_1}}, \mathbf{hv_{LBP_2}}, \cdots, \mathbf{hv_{LBP_{2^l}}}\}$ in the IM$_1$ are generated to represent all the possible 2^l LBP codes. Also, generate N hypervectors $\mathbf{Ch}_j$ in the IM$_2$ corresponding to the j^{th} channel, where $1 \leq j \leq N$. (a) Within a window, every $(l + 1)$ samples colored cyan in a certain channel can be extracted as an l-bit LBP code and encoded as a hypervector $\bar{\mathbf{hv}}_{\mathbf{LBP}_i}$, which is taken from $\{\mathbf{hv}_{\mathbf{LBP_1}}, \mathbf{hv}_{\mathbf{LBP_2}}, \cdots, \mathbf{hv}_{\mathbf{LBP}_{\mathbf{2}^l}}\}$ in the IM$_1$ based on its specific LBP code value. (b) Then the spatial LBP information for all N channels in the red box is represented by $\mathbf{hv}_{\mathbf{spatial}_i}$ in Eq. (3.1a), where $1 \leq i \leq W - l$. (c) Therefore, the k^{th} window colored blue is encoded by a hypervector $\mathbf{hv}_{\mathbf{win\_}k}$ as it adds all $W - l$ spatial information in Eq.(3.1b). (d) Finally, a class hypervector $\mathbf{hv}_{\mathbf{class}}$ can be formed by adding all K windows' information with majority rule as shown in Eq. (3.1c). (e) During the training phase, two class hypervectors for ictal and interictal states are trained by using Eq. (3.1a)-(3.1c). During the inference phase, a query hypervector $\mathbf{hv}_{\mathbf{query}}$ is generated by using Eq. (3.1a)-(3.1b). Similarity measurement is performed between the query hypervector and the trained class hypervectors to assign the predicted label.

3.2.2 PSD Method

Motivated by [64], where PSD achieves high classification performance on the Kaggle contest dataset using classification and regression tree (CART) based feature selection for polynomial SVM classifier. In this chapter, we use HDC with a selected small number of PSD features as well as using all PSD features.

All the iEEG data are preprocessed to extract band powers and remove power line noise [64]. (a) For dog subjects, the frequency band is split into 10 frequency sub-bands (Hz): 3-8, 8-13, 13-30, 30-55, 55-80, 80-105, 105-130, 130-150, 150-170, 170-200. (b) For human subjects, the frequency band is split into 13 frequency sub-bands (Hz): 3-8, 8-13, 13-30, 30-50, 50-80, 80-100, 100-130, 130-160, 160-200, 200-250, 250-300, 300-350, 350-400. To eliminate power line hums at 60 Hz and its harmonics, spectral powers in the band of $[60i{-}3, 60i{+}3]$ Hz are excluded in the PSD computation, where $i \in [0, 6]$. Note that Patient_1 is addressed as dogs with 10 frequency subbands since $f_s = 500$ Hz does not support the frequency subbands higher than 250 Hz.

Three spectral power metrics are computed as shown in Eq. (3.3), where ASP$_{f_1,f_2}$ refers to the absolute spectral power of a signal in the frequency band $[f_1, f_2]$ Hz, RSP$_{f_1,f_2}$ refers to the relative spectral power in band $[f_1, f_2]$ Hz and Ratio$_{f_1,f_2,f_3,f_4}$ represents the spectral power ratio of the absolute spectral power in band $[f_1, f_2]$ Hz over that in band $[f_3, f_4]$ Hz.

$$\mathrm{ASP}_{f_1,f_2} = \log \textstyle\sum_{f\in[f_1,f_2]} \mathrm{PSD}(f) \tag{3.3a}$$

$$\mathrm{RSP}_{f_1,f_2} = \log \frac{\sum_{f\in[f_1,f_2]} \mathrm{PSD}(f)}{\sum_{\text{all } f} \mathrm{PSD}(f)} \tag{3.3b}$$

$$\mathrm{Ratio}_{f_1,f_2,f_3,f_4} = \mathrm{ASP}_{f_1,f_2} - \mathrm{ASP}_{f_3,f_4} \tag{3.3c}$$

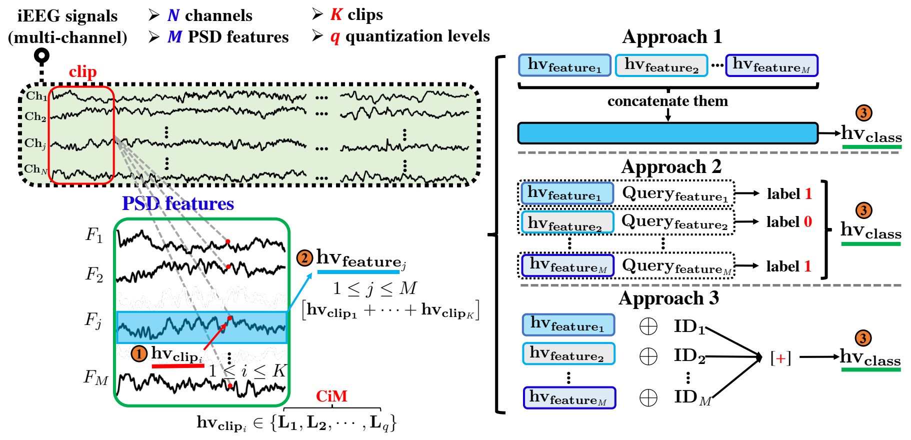


Figure 3.2: HD classification using PSD method.

Three Approaches for PSD Method

As shown in Figure 3.2, a certain class has N-channel iEEG signals with K clips. From each clip, M PSD feature values are extracted. These features are scaled into the range $[0, 1]$. This scaling step facilitates the encoding process for HDC. How to generate a *class* hypervector $\mathbf{hv_{class}}$ based on K clips with M PSD features needs to be addressed. Three novel approaches to solve this problem are described below and their HD classification approaches are shown in Figure 3.2.

Approach 1: Single Classifier, Long Hypervectors Before training, quantize the range $[0, 1]$ into q levels, a total q level hypervectors $\{\mathbf{L_1}, \mathbf{L_2}, \cdots, \mathbf{L}_q\}$ are generated in the CiM. As shown in Figure 3.2, for the j^{th} feature, clip i corresponds to a red data point and then should be represented by a *clip* hypervector $\mathbf{hv}_{\mathbf{clip}_i}$ based on its quantized level, where $\mathbf{hv}_{\mathbf{clip}_i} \in \{\mathbf{L_1}, \mathbf{L_2}, \cdots, \mathbf{L}_q\}$ and $i \in [1, N]$. As shown in Eq. (3.4), in total M *feature* hypervectors $\mathbf{hv}_{\mathbf{feature}_j}$ should be trained by adding all corresponding clip hypervectors, where $j \in [1, M]$ and $i \in [1, K]$. The final class hypervector is generated by concatenating all feature hypervectors. For example, if Dog_1 has 3 features, and each feature hypervector $\mathbf{hv}_{\mathbf{feature}_j}$ has its dimensionality $d = 10,000$, then the dimensionality for $\mathbf{hv}_{\mathbf{class}}$ becomes $30,000$.

$$\mathbf{hv}_{\mathbf{feature}_j} = \big[\mathbf{hv}_{\mathbf{clip_1}} + \cdots + \mathbf{hv}_{\mathbf{clip}_i} + \cdots + \mathbf{hv}_{\mathbf{clip}_K}\big], \tag{3.4a}$$

$$\mathbf{hv}_{\mathbf{class}} = (\mathbf{hv}_{\mathbf{feature_1}}, \cdots, \mathbf{hv}_{\mathbf{feature}_M}). \tag{3.4b}$$

Approach 2: Multiple Classifiers Approach 2 uses M feature hypervectors $\mathbf{hv}_{\mathbf{feature}_j}$ that are input to M classifiers, where $j \in [1, M]$. This is different from Approach 1 which employs a single long class hypervector $\mathbf{hv}_{\mathbf{class}}$. Therefore, a given test segment will produce M query hypervectors. Similarity measurement should be performed for each query hypervector. We obtain the label results from M classifiers. The final label is determined by a majority vote of all label results. For example, Patient_3 needs 4 features to determine the label. In this case, 4 feature hypervectors are trained. If the label 0 for interictal class is assigned 3 times, and 1 for ictal class is assigned once, then the final label should be classified as 0, namely the interictal segment. If the number of labels is even and half the labels correspond to each class, the tie is broken in favor of detection.

Approach 3: Single Classifier, Short Hypervectors Instead of concatenating the feature hypervectors, the hypervector $\mathbf{hv}_{\mathbf{class}}$ in Approach 3 is generated by Eq. (3.5), where the hypervectors' index ($\mathbf{ID}$) values are pre-generated in IM, whose total number is the same as selected features.

$$\mathbf{hv}_{\mathbf{clip}_i} \in \{\mathbf{L_1}, \mathbf{L_2}, \cdots, \mathbf{L}_q\}, \text{ where } i \in [1, K], \tag{3.5a}$$

$$\mathbf{hv}_{\mathbf{feature}_j} = \big[\mathbf{hv}_{\mathbf{clip_1}} + \cdots + \mathbf{hv}_{\mathbf{clip}_K}\big], \tag{3.5b}$$

$$\mathbf{hv}_{\mathbf{class}} = \big[\mathbf{hv}_{\mathbf{feature_1}} \oplus \mathbf{ID_1} + \cdots + \mathbf{hv}_{\mathbf{feature}_M} \oplus \mathbf{ID}_M\big]$$

Selected and All PSD Features

Selected PSD Features Inspired by [56], we use the Fisher score to select efficient features for discrimination [111]. These features are linearly separable. The higher the Fisher score, the more linearly separable the feature is. Top *three* PSD features selected using Fisher score are shown in Table 3.1.

Table 3.1: Three selected features based on Fisher score.

Patient	**Selected Features**
Dog_1	ele_{12} : $\text{ASP}_{170,200}$, $\text{ASP}_{150,170}$ ele_{16} : $\text{ASP}_{170,200}$
Dog_2	ele_{16} : $\text{Ratio}_{3,8,80,105}$ ele_{14} : $\text{Ratio}_{3,8,80,105}$ ele_{11} : $\text{Ratio}_{3,8,80,105}$
Dog_3	ele_{7} : $\text{ASP}_{13,30}$ ele_{14} : $\text{ASP}_{13,30}$ ele_{13} : $\text{ASP}_{13,30}$
Dog_4	ele_{6} : $\text{ASP}_{3,8}$ ele_{12} : $\text{Ratio}_{80,105,150,170}$, $\text{Ratio}_{55,80,150,170}$
Patient_1	ele_{19} : $\text{ASP}_{80,105}$, $\text{ASP}_{55,80}$ ele_{10} : $\text{Ratio}_{8,13,80,105}$
Patient_2	ele_{1} : $\text{ASP}_{13,30}$ ele_{3} : $\text{ASP}_{13,30}$ ele_{2} : $\text{ASP}_{13,30}$
Patient_3	ele_{6} : $\text{ASP}_{13,30}$ ele_{5} : $\text{ASP}_{13,30}$, $\text{ASP}_{8,13}$
Patient_4	ele_{44} : $\text{Ratio}_{80,100,130,160}$ ele_{70} : $\text{Ratio}_{300,350,350,400}$, $\text{Ratio}_{250,300,350,400}$
Patient_5	ele_{1} : $\text{ASP}_{160,200}$ ele_{25} : $\text{ASP}_{30,50}$ ele_{2} : $\text{ASP}_{50,80}$
Patient_6	ele_{24} : $\text{Ratio}_{13,30,250,300}$, $\text{Ratio}_{13,30,200,250}$ $\text{Ratio}_{13,30,300,350}$
Patient_7	ele_{28} : $\text{ASP}_{3,8}$, $\text{Ratio}_{13,30,350,400}$, $\text{ASP}_{13,30}$
Patient_8	ele_{10} : $\text{ASP}_{13,30}$, $\text{ASP}_{8,13}$ ele_{11} : $\text{ASP}_{13,30}$

All PSD Features We also take all features into consideration, which means all three spectral power metrics over all sub-bands are computed. Therefore, (a) for dog subjects who have 10 sub-bands, a total of $65(= 10 + 10 + \binom{10}{2})$ features need to be computed. They are 10 ASP, 10 RSP and $\binom{10}{2}$ Ratio. (b) Similarly, for human subjects with 13 sub-bands, $104(= 13 + 13 + \binom{13}{2})$ features are computed. Similar to a small number of PSD features method, three approaches are performed for all features with HDC. The only difference is the generation of $\mathbf{hv_{class}}$ hypervector in Approach 3, which is shown in Eq. (3.6). The algorithm is straightforward: (a) Before training, pre-generate N channel hypervectors $\mathbf{Ch}_k$ in IM$_1$, M ID hypervectors $\mathbf{ID}_j$ in IM$_2$, and q level hypervectors $\{\mathbf{L_1}, \mathbf{L_2}, \cdots, \mathbf{L}_q\}$ in CiM, where $k \in [1, N]$ and $j \in [1, M]$. (b) For each clip, generate the clip hypervector as described in Eq. (3.6a), where $[\cdot]$ represents the majority rule and $\bar{\mathbf{L}}_j \in \{\mathbf{L_1}, \cdots, \mathbf{L}_q\}$. The final class hypervector is generated by adding all clip hypervectors for the same class as shown in Eq. (3.6b). (c) Once the two class hypervectors are trained, during the testing phase, the query hypervector is generated by each clip in the same way as shown in Eq. (3.6a). Finally, the label is assigned according to the similarity measurement between the query hypervectors and the trained two class hypervectors.

$$\begin{aligned}\mathbf{hv}_{\mathbf{clip}_i} = \big[&\big[\mathbf{ID_1} \oplus \bar{\mathbf{L}}_\mathbf{1} + \cdots + \mathbf{ID}_M \oplus \bar{\mathbf{L}}_M\big] \oplus \mathbf{Ch_1} + \cdots \\ &+ \big[\mathbf{ID_1} \oplus \bar{\mathbf{L}}_\mathbf{1} + \cdots + \mathbf{ID}_M \oplus \bar{\mathbf{L}}_M\big] \oplus \mathbf{Ch}_N\big] \qquad (3.6a)\end{aligned}$$

$$\mathbf{hv_{class}} = \big[\mathbf{hv_{clip_1}} + \mathbf{hv_{clip_2}} + \cdots + \mathbf{hv}_{\mathbf{clip}_K}\big] \qquad (3.6b)$$

3.2.3 Hypervector Distance Plot

In this chapter, a *hypervector distance* plot is introduced to visualize the scatter plot of classification results. As an example, Figure 3.3, a hypervector distance plot illustrates the Hamming distance between the ictal class hypervector $\mathbf{hv_{ictal}}$ and the query hypervector $\mathbf{hv_{query}}$ as its x-axis, and the Hamming distance between the interictal class hypervector $\mathbf{hv_{interictal}}$ and the query hypervector $\mathbf{hv_{query}}$ on the y-axis. A blue line, which represents a set of query hypervectors that have the same distance with ictal and interictal hypervectors, is introduced in this plot. Additionally, ictal hypervectors are shown as the red points, while the black points represent the interictal hypervectors.

For high classification accuracy, both red and blue points should be located farther from the blue line. Generally, for a binary classifier with 100% test accuracy, all red points are above the blue line, and all black points are below the blue line.

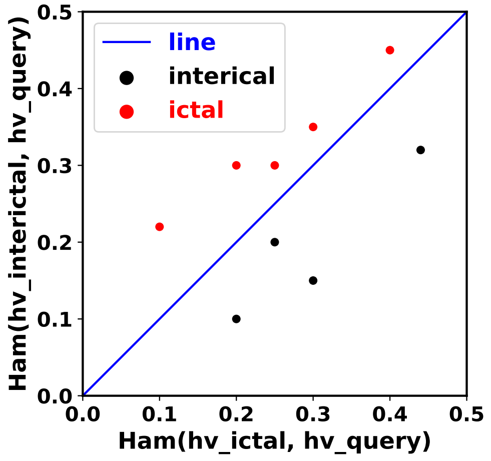


Figure 3.3: A general hypervector distance plot for an ideal binary classification.

To the best of our knowledge, we are the first to introduce the hypervector distance plot for HD classifiers to visualize the accuracy of binary classification. Based on the corresponding plots in Sec. 3.4.3, we find these plots can also display the classification performance for different classifiers. More details are discussed in Sec. 3.4.3.

3.3 Materials

3.3.1 iEEG Dataset from Kaggle Contest

The dataset for testing the proposed algorithm is from the UPenn and Mayo Clinic's Seizure Detection Challenge [28] organized by Kaggle [110]. The experimental procedures involving human subjects described in this chapter were approved by the Institutional Review Board. The experimental procedures involving animal models described in this chapter were approved by the Institutional Animal Care and Ethics Committee.

Table 3.2 lists the number of ictal and interictal clips in the training set, the number of test clips, the number of channels and the sampling frequency of the iEEG dataset from the Kaggle seizure detection contest [28], where each clip represents a one second

Table 3.2: Dataset information.

Patient	**#ictal**	**#interictal**	**#test**	**#channel**	f_s **(Hz)**
Dog_1	178	418	3181	16	400
Dog_2	172	1148	2997	16	400
Dog_3	480	4760	4450	16	400
Dog_4	257	2790	3013	16	400
Patient_1	70	104	2050	68	500
Patient_2	151	2990	3894	16	5000
Patient_3	327	714	1281	55	5000
Patient_4	20	190	543	72	5000
Patient_5	135	2610	2986	64	5000
Patient_6	225	2772	2997	30	5000
Patient_7	282	3239	3601	36	5000
Patient_8	180	1710	1922	16	5000

iEEG data. This contest analyzes iEEG data from 4 dogs and 8 human subjects. The sampling frequency f_s of these recordings is 400 Hz for dogs, and is 500 Hz or 5000 Hz for human subjects. Interested readers are referred to [110] for more details.

3.3.2 Training, Validation and Test Data

The data in the Kaggle contest has already been segmented into training and test clips. Training clips are arranged sequentially, whereas test data are labelled randomly. This indicates that the test clips labelled consecutively are not taken from the continuous time series.

To avoid model overfitting, validation data is split from the training data. Since the dynamics of seizures are different even for a single subject, similar to [112], we conduct the leave-one-seizure-out cross-validation (LOOCV) per subject. Specifically, based on the number of seizures #Sz, the original training data is divided into #Sz folds with the interictal clips divided equally. Each fold contains one seizure. Then the HD models are trained over all but one (or $\#\text{Sz} - 1$) folds, validate over the held-out one fold and test over the given test data. The final evaluation results are the average of all cross-validation rounds.

3.3.3 Performance Evaluation

As a binary classification problem, seizure detection results in four possible outcomes: true positive (TP), true negative (TN), false positive (FP) and false negative (FN).

Table 3.3: Performance metrics.

Metric	Formula	Expected
Accuracy	$\frac{\text{TP+TN}}{\text{TP+TN+FP+FN}}$	high
Sensitivity	$\frac{\text{TP}}{\text{TP+FN}}$	high
Specificity	$\frac{\text{TN}}{\text{TN+FP}} = 1 - \text{FPR}^1$	high
AUC	area under the ROC curve ($\in [0,1]$)	high
latency	$\Delta = t_{\text{seizure is detected}} - t_{\text{actual seizure onset}}$	low

[1] FPR represents the false positive rate.

These quantities are used to evaluate the metrics—accuracy, sensitivity and specificity—to measure the performance of the detection algorithms. Accuracy reveals an overall classification performance in seizure detection. Sensitivity, also referred as true positive rate (TPR), reflects how well the seizure (ictal/positive) data is correctly detected, which is the ratio of correctly classified ictal labels to the total ictal data. Specificity, also named true negative rate (TNR), indicates how well the normal baseline (interictal/false) data is correctly classified. Due to the nature of epilepsy, where the seizure duration typically lasts from ten seconds to two minutes, seizure detection is essentially an imbalanced data classification problem. Accordingly, accuracy cannot sufficiently measure the detection performance for highly imbalanced datasets. The receiver operating characteristic (ROC) curve is a plot of "sensitivity" versus "1-specificity", and the area under this curve (AUC) can be used to measure the overall classification performance. In this chapter, we use test accuracy, sensitivity, specificity, and AUC to measure the classification performance, where AUC is the main metric to select the best seizure detection algorithm. Other than the classification performance, seizure detection also considers how fast the proposed algorithm can detect the seizure. To indicate this, the metric *latency* is denoted as the time difference between the actual seizure onset and the detection time determined by the algorithm. More details for the

3.4 Experimental Results

3.4.1 LBP Method

Using one-second window length, experiments are conducted for different LBP code lengths.

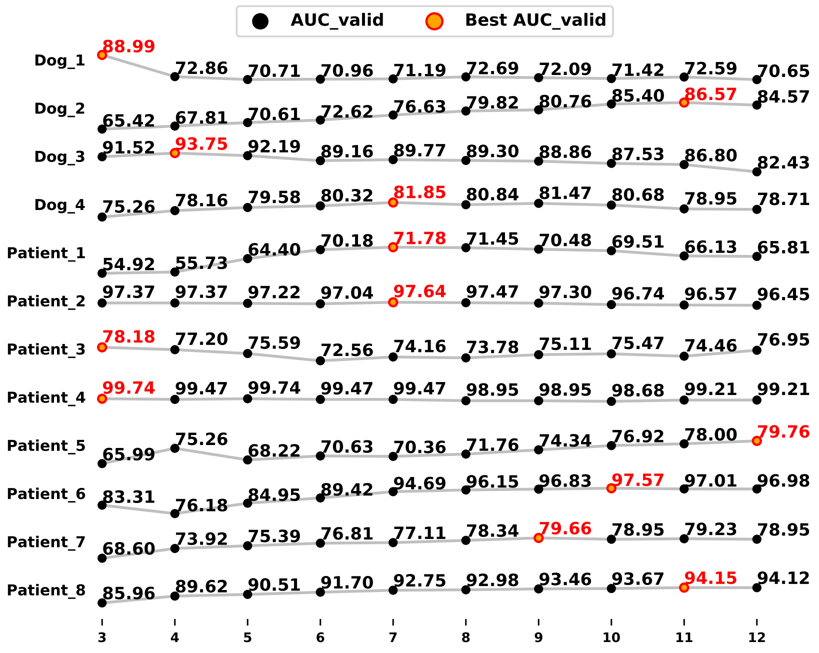


Figure 3.4: AUC for validation data.

Table 3.4: Average LBP validation AUC performances.

l	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Ave_1[1]	75.22	73.66	75.50	76.65	78.25	78.82	78.73	**78.91**	78.21	76.44
Ave_2[2]	82.74	84.15	84.52	85.37	86.60	87.06	87.95	88.28	88.38	**88.92**
Ave_3[3]	79.61	79.78	80.76	81.74	83.12	83.63	84.11	**84.38**	84.14	83.72

[1] Ave_1: average AUC performance for four dogs and Patient_1.
[2] Ave_2: average AUC performance for Patient_2 to Patient_8.
[3] Ave_3: average AUC performance for twelve subjects.

Simulation results for AUC over validation data are shown in Figure 3.4, where the x-axis for each figure denotes different LBP code lengths, with the corresponding summary listed in Table 3.4. From this table, Ave_1 denotes the average AUC performance for the subjects with $f_s \leq$ 500Hz, Ave_2 is the average AUC performance for patients with f_s=5kHz and Ave_3 is the average AUC performance for all twelve subjects. In terms of the validation data, patients with f_s=5kHz on average achieve the best AUC of 88.92% when l=12, whereas subjects with $f_s \leq$500Hz have the best AUC of 78.91% when l=10. Note that, indicated by Ave_2, the average AUC performance when l=10 is about the same as that of the best performance with l=12. Using $l = 6$ reduces AUC about 2% compared to $l = 10$. The performance results over test data using the LBP method are shown in Tables 3.7 and 3.8 with a detailed discussion later in Sec. 3.4.3.

Table 3.5: Performance for selected PSD features using fisher score with quantization level $q = 21$.[1]

Patient	**Approach 1**							**Approach 2**							**Approach 3**							**Best Approach**
	A$_{test}$	**Sen.**	**Spec.**	**AUC test**	**AUC valid**	**AUC train**	**Lat.**	**A$_{test}$**	**Sen.**	**Spec.**	**AUC test**	**AUC valid**	**AUC train**	**Lat.**	**A$_{test}$**	**Sen.**	**Spec.**	**AUC test**	**AUC valid**	**AUC train**	**Lat.**	
Dog_1	95.6	91.8	95.8	93.8	97.9	98.0	0.8	95.6	92.3	95.8	**94.0**	98.0	98.0	0.8	95.6	91.4	95.8	93.6	97.8	98.0	0.8	SF_v2
Dog_2	86.9	94.5	86.4	**90.5**	94.2	94.7	0.7	86.0	93.0	85.7	89.3	94.2	94.1	0.7	87.8	86.6	87.9	87.2	93.9	94.8	0.3	SF_v1
Dog_3	95.3	72.5	97.5	85.0	92.0	92.5	3.2	95.0	73.7	97.1	**85.4**	91.8	92.4	3.2	95.3	72.1	97.6	84.8	91.7	92.1	3.1	SF_v2
Dog_4	64.9	83.2	64.1	**73.7**	81.8	82.2	0.0	63.4	83.2	62.5	72.9	80.8	82.1	0.0	64.6	80.4	63.9	72.1	82.0	82.6	0.0	SF_v1
Patient_1	95.5	94.2	95.7	94.9	91.7	92.8	2.0	93.9	96.6	93.6	**95.1**	88.9	94.9	2.0	94.6	92.6	94.7	93.7	91.3	93.9	1.8	SF_v2
Patient_2	98.1	95.3	98.2	96.8	96.1	95.9	0.0	97.7	93.9	98.0	95.9	95.2	95.2	0.0	98.1	95.3	98.3	**96.8**	95.6	96.0	0.0	SF_v3
Patient_3	95.0	78.5	96.9	87.7	82.7	79.6	6.9	94.7	79.0	96.4	**87.7**	82.2	79.4	6.9	94.7	75.9	96.8	86.4	82.6	79.4	6.9	SF_v2
Patient_4	87.6	21.0	94.3	57.7	97.4	97.4	0.0	87.0	25.0	93.3	59.2	97.4	97.4	0.0	88.2	35.0	93.6	**64.3**	97.4	97.4	0.0	SF_v3
Patient_5	85.2	91.3	84.9	**88.1**	79.3	81.9	6.7	84.3	90.9	83.9	87.4	80.2	81.6	6.7	82.9	90.7	82.4	86.5	79.0	81.8	6.8	SF_v1
Patient_6	96.8	60.9	99.6	80.3	95.4	95.4	1.5	97.1	66.4	99.6	**83.0**	96.7	96.7	1.5	96.8	60.9	99.6	80.3	95.4	95.4	1.5	SF_v2
Patient_7	98.1	83.1	99.8	**91.4**	76.9	81.5	28.7	97.6	82.3	99.4	90.8	76.2	81.5	28.7	98.1	82.5	99.8	91.2	76.7	81.4	29.4	SF_v1
Patient_8	94.1	91.7	94.3	93.0	95.7	95.7	3.5	93.6	91.7	93.8	92.7	95.5	95.4	3.5	94.4	91.7	94.7	**93.2**	95.8	95.8	3.5	SF_v3
mean	91.1	79.8	92.3	86.1	90.1	90.6	4.5	90.5	80.7	91.6	86.1	89.8	90.7	4.5	90.9	79.6	92.1	85.8	89.9	90.7	4.5	/

[1] A$_{test}$: test accuracy (%), Sen.: sensitivity (%), Spec.: Specificity (%), Lat.: latency (s), SF_vi: Approach i for selected features, where $i \in [1, 2, 3]$.

3.4.2 PSD Method

PSD Method using Selected Features

Simulation results are shown in Table 3.5 using three approaches. In Table 3.5, A_{test} represents the accuracy on the testing data. The best AUC performance over test data for each subject is highlighted in bold. Simulations show that: Indicated by the test AUC, all these three approaches achieve a similar overall detection performance with 91% test accuracy, 80% sensitivity, 92% specificity, 86% test AUC and 4.5s latency.

Table 3.6: Performance for all PSD features with quantization level $q = 21$.[1,2]

Patient	Approach 1							Approach 2							Approach 3							Best
	A_{test}	**Sen.**	**Spec.**	**AUC_{test}**	**AUC_{valid}**	**AUC_{train}**	**Lat.**	**A_{test}**	**Sen.**	**Spec.**	**AUC_{test}**	**AUC_{valid}**	**AUC_{train}**	**Lat.**	**A_{test}**	**Sen.**	**Spec.**	**AUC_{test}**	**AUC_{valid}**	**AUC_{train}**	**Lat.**	**Approach**
Dog_1	99.5	91.9	99.9	**95.9**	95.2	97.4	1.0	75.8	98.0	74.6	86.3	80.4	83.5	0.4	99.0	87.7	99.5	93.6	94.8	94.2	1.6	AF_v1
Dog_2	91.1	85.7	91.4	**88.6**	90.8	91.5	0.8	87.9	88.2	87.9	88.0	90.9	91.1	0.2	84.9	89.5	84.6	87.1	90.5	92.6	0.4	AF_v1
Dog_3	95.8	88.1	96.5	**92.3**	86.1	87.7	3.8	64.2	96.4	61.0	78.7	78.4	79.1	0.0	92.6	90.7	92.8	91.8	88.7	89.1	2.7	AF_v1
Dog_4	64.7	74.0	64.3	69.1	75.7	76.5	4.3	49.5	91.2	47.7	**69.4**	78.6	77.6	0.3	67.7	57.6	68.1	62.9	76.8	78.2	1.3	AF_v2
Patient_1	83.8	20.2	89.3	54.8	74.8	91.8	8.5	74.7	42.9	77.5	60.2	64.6	87.9	7.0	88.2	46.0	91.8	**68.9**	72.7	91.9	7.3	AF_v3
Patient_2	71.4	63.8	71.9	67.8	56.6	68.8	2.4	26.0	95.3	21.6	58.5	55.7	58.5	0.3	89.9	77.3	90.7	**84.0**	80.0	83.3	0.7	AF_v3
Patient_3	36.4	57.7	34.0	45.9	57.3	62.8	0.2	11.1	97.2	1.5	**49.4**	48.6	51.1	0.0	41.5	52.5	40.3	46.4	57.9	68.4	1.3	AF_v2
Patient_4	52.1	15.0	55.9	35.4	73.2	85.9	0.0	44.2	19.0	46.8	32.9	71.8	83.6	0.0	68.6	14.0	74.1	**44.1**	84.3	98.7	0.3	AF_v3
Patient_5	79.0	84.5	78.7	81.6	79.1	78.8	13.8	65.2	85.9	64.0	74.9	77.3	76.2	13.7	84.5	84.9	84.5	**84.7**	79.9	80.6	13.8	AF_v3
Patient_6	90.0	45.0	93.6	69.3	86.7	89.8	2.5	46.8	84.3	43.9	64.1	69.3	71.0	0.6	93.6	66.4	95.7	**81.0**	94.2	93.7	1.3	AF_v3
Patient_7	88.1	76.1	89.4	82.7	69.1	77.6	5.1	57.6	89.5	54.1	71.8	58.6	66.9	4.9	90.5	83.2	91.3	**87.3**	71.3	82.4	4.6	AF_v3
Patient_8	78.1	81.7	77.7	**79.7**	70.1	89.0	4.3	31.3	95.8	24.6	60.2	53.4	64.6	0.0	78.5	76.9	78.7	77.8	70.9	88.4	3.3	AF_v1
mean	77.5	65.3	78.5	71.9	76.2	83.1	3.9	52.9	82.0	50.4	66.2	69.0	74.3	2.3	81.6	68.9	82.7	75.8	80.2	86.8	3.2	/

[1] A_{test}: test accuracy (%), Sen.: sensitivity (%), Spec.: Specificity (%), Lat.: latency (s), AF_vi: Approach i for all features, where $i \in [1, 2, 3]$.
[2] The results are different from [20] where the normalization should have been done by merging ictal and interictal data.

Table 3.7: Summary for LBP and PSD methods.

Patient	LBP Method						PSD Method (Selected Features)						PSD Method (All Features)					
	A_{test}	**Sen.**	**Spec.**	**AUC_{test}**	**Lat.**	l	**A_{test}**	**Sen.**	**Spec.**	**AUC_{test}**	**Lat.**	**Best**	**A_{test}**	**Sen.**	**Spec.**	**AUC_{test}**	**Lat.**	**Best**
Dog_1	94.2	71.9	95.3	83.6	1.7	3	95.6	92.3	95.8	94.0	0.8	SF_v2	99.5	91.9	99.9	**95.9**	1.0	AF_v1
Dog_2	77.9	57.7	79.0	68.3	0.0	11	86.9	94.5	86.4	**90.5**	0.7	SF_v1	91.1	85.7	91.4	88.6	0.8	AF_v1
Dog_3	92.3	82.9	93.2	88.1	1.8	4	95.0	73.7	97.1	85.4	3.2	SF_v2	95.8	88.1	96.5	**92.3**	3.8	AF_v1
Dog_4	61.6	54.0	61.9	58.0	6.8	7	64.9	83.2	64.1	**73.7**	0.0	SF_v1	49.5	91.2	47.7	69.4	0.3	AF_v2
Patient_1	88.2	20.2	94.0	57.1	6.0	7	93.9	96.6	93.6	**95.1**	2.0	SF_v2	88.2	46.0	91.8	68.9	7.3	AF_v3
Patient_2	98.0	95.6	98.1	**96.9**	0.0	7	98.1	95.3	98.3	96.8	0.0	SF_v3	89.9	77.3	90.7	84.0	0.7	AF_v3
Patient_3	11.2	100.0	1.3	50.7	2.0	3	94.7	79.0	96.4	**87.7**	6.9	SF_v2	11.1	97.2	1.5	49.4	0.0	AF_v2
Patient_4	80.8	51.0	83.9	**67.4**	0.0	3	88.2	35.0	93.6	64.3	0.0	SF_v3	68.6	14.0	74.1	44.1	0.3	AF_v3
Patient_5	93.2	88.1	93.5	**90.8**	6.3	12	85.2	91.3	84.9	88.1	6.7	SF_v1	84.5	84.9	84.5	84.7	13.8	AF_v3
Patient_6	97.8	73.3	99.8	**86.5**	1.6	10	97.1	66.4	99.6	83.0	1.5	SF_v2	93.6	66.4	95.7	81.0	1.3	AF_v3
Patient_7	77.4	75.6	77.6	76.6	0.2	9	98.1	83.1	99.8	**91.4**	28.7	SF_v1	90.5	83.2	91.3	87.3	4.6	AF_v3
Patient_8	98.3	92.2	98.9	**95.6**	3.8	11	94.4	91.7	94.7	93.2	3.5	SF_v3	78.1	81.7	77.7	79.7	4.3	AF_v1
mean_1[2]	80.9	71.9	81.4	76.6	2.5	/	91.0	81.8	92.0	**86.9**	4.5	/	78.4	75.6	78.6	77.1	3.1	/
mean_2[3]	79.3	71.8	79.6	75.7	3.1	10	90.9	79.6	92.1	85.8	4.5	SF_v3	81.6	68.9	82.7	75.8	3.2	AF_v3
[109]	LBP method using HD						95.4	96.0	94.8	/	15.9	/	SWEC-ETHZ iEEG dataset[4]					
[64]	PSD Method using SVM classifiers[5]						/	100.0	99.9	/	5.8	/	Kaggle dataset					

[1] A_{test}: test accuracy (%), Sen.: sensitivity (%), Spec.: Specificity (%), SF: selected features, AF: all features, vi: Approach i, where $i \in [1, 2, 3]$, Lat.: latency with the unit of seconds.
[2] mean_1: average performance for each subject with the highest test AUC.
[3] mean_2: average performance for all subjects using one method.
[4] SWEC-ETHZ iEEG dataset with only training data available.
[5] PSD features are selected using CART. No HDC is used.

Table 3.8: Impact of hypervector sizes for the test AUC using LBP and PSF methods.

	LBP		**PSD Method (Selected Features)**									**PSD Method (All Features)**								
Patient	$l = 10$		**Approach 1**			**Approach 2**			**Approach 3**			**Approach 1**			**Approach 2**			**Approach 3**		
	10*k*	**1*k***	**10*k***	**1*k***	**100**	**10*k***	**1*k***	**100**	**10*k***	**1*k***	**100**	**10*k***	**1*k***	**100**	**10*k***	**1*k***	**100**	**10*k***	**1*k***	**100**
Dog_1	77.0	79.1	93.8	93.8	93.8	94.0	94.0	94.0	93.6	94.0	94.9	95.9	95.9	95.9	86.3	86.3	81.3	93.6	78.9	54.7
Dog_2	66.8	66.3	90.5	90.5	90.5	89.3	89.3	89.3	87.2	87.2	90.3	88.6	88.5	88.5	88.0	87.7	87.3	87.1	86.1	69.7
Dog_3	81.0	76.1	85.0	85.0	85.0	85.4	85.4	85.4	84.8	84.4	85.7	92.3	92.3	92.3	78.7	78.5	75.8	91.8	85.1	66.3
Dog_4	58.3	62.0	73.7	73.7	73.7	72.9	72.9	72.9	72.1	74.8	69.7	69.1	69.1	68.7	69.4	69.4	68.6	62.9	70.2	46.8
Patient_1	59.7	59.7	94.9	94.9	94.9	95.1	95.1	95.1	93.7	92.2	80.8	54.8	54.5	54.4	60.2	60.4	62.8	68.9	60.2	56.0
Patient_2	96.5	96.6	96.8	96.8	96.8	95.9	95.9	95.9	96.8	96.8	96.7	67.8	68.0	67.7	58.5	58.4	52.7	84.0	64.6	59.1
Patient_3	55.9	54.2	87.7	87.7	87.7	87.7	87.7	87.7	86.4	87.1	86.6	45.9	45.9	45.9	49.4	49.4	49.4	46.4	49.5	52.7
Patient_4	63.4	64.9	57.7	57.7	57.7	59.2	58.7	58.7	64.3	65.9	60.0	35.4	35.3	36.1	32.9	32.5	31.3	44.1	45.3	51.8
Patient_5	90.1	89.6	88.1	87.8	87.8	87.4	87.4	86.9	86.5	86.8	83.1	81.6	81.6	81.7	74.9	74.8	74.2	84.7	74.7	60.8
Patient_6	86.5	87.0	80.3	80.3	80.3	83.0	83.0	83.0	80.3	80.3	81.0	69.3	69.5	69.4	64.1	63.9	59.7	81.0	65.6	55.4
Patient_7	78.3	78.2	91.4	91.4	91.4	90.8	90.8	90.8	91.2	91.6	91.0	82.7	82.7	82.9	71.8	71.7	70.1	87.3	68.0	57.1
Patient_8	95.0	94.2	93.0	93.0	93.0	92.7	92.7	92.7	93.2	93.2	92.1	79.7	79.4	80.4	60.2	60.4	59.3	77.8	70.1	53.8
mean	75.7	75.7	86.1	86.0	86.0	86.1	86.1	86.0	85.8	86.2	84.3	71.9	71.9	72.0	66.2	66.1	64.4	75.8	68.2	57.0

Table 3.9: Memory and computational requirements for LBP and PSD methods.[1]

Type[2]	**SF_v1/v2**	**SF_v3**	**AF_v1/v2**	**AF_v3**	**LBP**
IM and CiM	q	$q+3$	q	$q+M+N$	2^l+N
AM	$6(=2*3)$	2	$2M$	2	2
total	$q+6$	$q+5$	$q+2M$	$q+M+N+2$	2^l+N+2
# addition	$3(K-1)$	$3(K-1)+2$	$M(K-1)$	$MNK-1$	$NK(f_s-l)-1$
# multiplication	/	3	/	$(M+1)NK$	$NK(f_s-l)$
total	$3(K-1)$	$3K+2$	$M(K-1)$	$(2M+1)NK-1$	$2NK(f_s-l)-1$

[1] q: the quantization level, M: number of all PSD features, where $M = 65$ for dogs and $M = 104$ for human subjects, N: number of channels, l: LBP-code length, K: number of segments or clips for a certain class, f_s: sampling frequency.

[2] SF: selected features, AF: all features, vi: Approach i, where $i \in [1, 2, 3]$.

Table 3.10: Specific memory requirements for LBP and PSD methods.

Patient	N	**PSD (selected)**		**PSD (all)**		**LBP method**									
		SF_v1,2	**SF_v3**	**AF_v1,2**	**AF_v3**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Dog1	16	27	**26**	151	104	**26**	34	50	82	146	274	530	1042	2066	4114
Dog2	16	27	**26**	151	104	**26**	34	50	82	146	274	530	1042	2066	4114
Dog3	16	27	**26**	151	104	**26**	34	50	82	146	274	530	1042	2066	4114
Dog4	16	27	**26**	151	104	**26**	34	50	82	146	274	530	1042	2066	4114
Patient1	68	27	**26**	229	195	78	86	102	134	198	326	582	1094	2118	4166
Patient2	16	27	**26**	229	143	**26**	34	50	82	146	274	530	1042	2066	4114
Patient3	55	27	**26**	229	182	65	73	89	121	185	313	569	1081	2105	4153
Patient4	72	27	**26**	229	199	82	90	106	138	202	330	586	1098	2122	4170
Patient5	64	27	**26**	229	191	74	82	98	130	194	322	578	1090	2114	4162
Patient6	30	27	**26**	229	157	40	48	64	96	160	288	544	1056	2080	4128
Patient7	36	27	**26**	229	163	46	54	70	102	166	294	550	1062	2086	4134
Patient8	16	27	**26**	229	143	**26**	34	50	82	146	274	530	1042	2066	4114

[1] N: number of channels, SF: selected features, AF: all features, vi: Approach i, where $i \in [1, 2, 3]$.

PSD Method using All Features

Simulation results for three approaches using all features are shown in Table 3.6: (a) Indicated by the average test AUC, Approach 3 achieves an overall best detection performance among all three approaches: 82% test accuracy, 70% sensitivity, 83% specificity, 76% test AUC and 3.2s latency. (b) For every subject, the best AUC performance over test data is highlighted in bold. Based on this table, Approach 1 is more suitable for dogs with three out of four cases achieving the best overall AUC performance, whereas Approach 3 is better suited for human subjects since six out of eight cases achieve the best performance over test data.

Comparison between Selected and All PSD Features

Comparing Tables 3.5 and 3.6, we can observe that, for all subjects except Dog_1 and Dog_3, using selected features can achieve higher performance than that using all features, which indicates that feature selection plays an important role in the field of HDC for classification. Additionally, though using all frequency-domain information does not improve the classification performance, it reduces the latency from 4.5s to 2.3s.

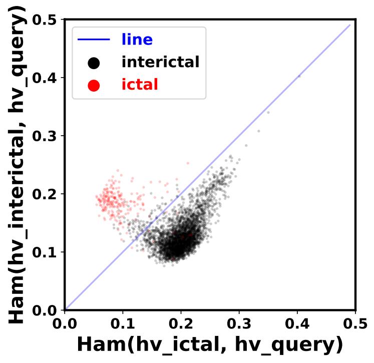
3.4.3 Discussion on LBP and PSD Methods

Table 3.7 summarizes the best LBP and PSD methods using HDC for seizure detection using the Kaggle dataset, and compares them with the previous work. It can be observed that: indicated by the average AUC performance over test data, PSD method using selected features achieves the best seizure detection performance, which can lead to an average performance of: 91% test accuracy, 82% sensitivity, 92% specificity, 87% test AUC and 4.5s latency. Additionally, though LBP method outperforms the PSD method for five human subjects, the test AUC performance difference is small. Compared to two previous work, we find: (a) The LBP method using HDC works nearly perfectly for the SWEC-ETHZ iEEG dataset [27] as claimed in [109], but not for the Kaggle dataset. The reason for this lower performance is twofold: (i) Post-processing techniques, such as k-out-of-n majority voting (= over k consecutive predictions for n-second window are classified correctly) [107, 113], are typically employed after the classifier to enhance the detection performance. For example, [114] reveals that the average accuracy for seizure prediction can be improved from 73.6% to 93.3% by applying post-processing. Note that each segment in SWEC-ETHZ iEEG dataset is an at least 6-min iEEG recordings that contain the whole seizure onset, whereas each segment (or clip) is a one-second iEEG recordings and arranged randomly for test in the Kaggle dataset. Therefore the post-processing employed in [109], which is essentially k-out-of-ten majority rule ($k \geq 7$), cannot be applied to the Kaggle dataset. (ii) SWEC-ETHZ iEEG dataset only offers the training data, while the Kaggle dataset offers both the training and the already held-out test data. Since the dynamics of different seizures for a certain subject vary widely, there may exist a big difference in the patterns between the held-out test data and the given training data. (b) Although the Kaggle dataset was used in [64], that paper uses the traditional SVM classifiers with feature selection; this requires more execution time in training and may consume more energy in real-time implementation.

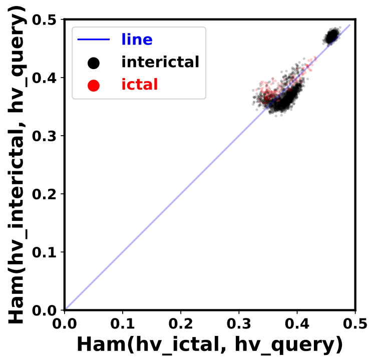
Table 3.8 lists the AUC performance over test data with the dimensionality d of hypervectors reduced from the default $10,000$ to $1,000$ or 100 for LBP and PSD methods. Compared to the baseline performance with $d = 10,000$, the detection performances, with an exception of the Approach 3 using all PSD features, are comparable for both LBP and PSD methods when d=$1,000$. For PSD methods, all but Approach 3 using all PSD features can achieve comparable performances even when d=100. This is because the capacity of the hypervectors with $d = 100$ is not sufficient to guarantee the required number of orthogonal hypervectors in IM for Approach 3 using all PSD features.

The memory and complexity requirements results are shown in Table 3.9, where the memory requirements for storing hypervectors are expressed as a factor of the dimensionality d of the hypervectors and complexity requirements are expressed as the number of arithmetic operations. Note that the cost of extracting LBP codes and PSD values is not considered. Both the memory and complexity requirements are different among approaches. They depend on the number of channels (N), features (M), quantization levels (q) for feature values, the LBP code length (l) and the sampling frequency (f_s). The specific memory requirements are shown in Table 3.10. (a) For the memory requirements, based on Tables 3.9 and 3.10, Approach 3 of the PSD method using selected features requires the minimal memory size. Both Approaches 1 and 2 of the PSD method using selected features consume the second minimal memory storage. The LBP method with the code length higher than 8 requires more memory storage than the PSD methods. (b) For the complexity requirements, the total number of the operations is listed in the last row of Table 3.9. Coincidentally, the order from the left to right reflects the number of operations from minimal to maximal. Therefore, Approaches 1 and 2 of the PSD method with selected features require the minimal number of operations. In summary, in terms of memory and complexity, the PSD method using selected features outperforms the LBP method.

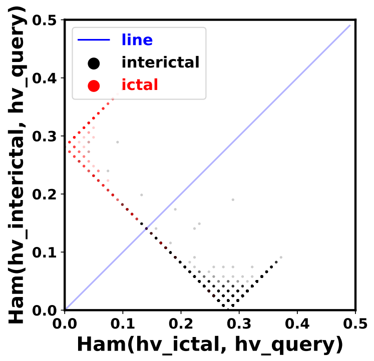
Figure 3.5 shows the hypervector distance plots for Patient_2 (left panels) and Dog_4 (right panels) among the proposed different HD classifiers using the model that achieves the highest validation AUC. Note that, no hypervector distance plot for selected or all features in the PSD method for Approach 2 is shown in the figure. By observation, for Patient_2, all HD classifiers except PSD method using all features can discriminate the two classes well, as data from two classes are located far from the blue line. For Dog_4, the two classes of data are not discriminable by any HD method; this can be observed from the plots in the right panel of Figure 3.5. This visualization indicates that feature selection plays an important role in the data separability for HDC.



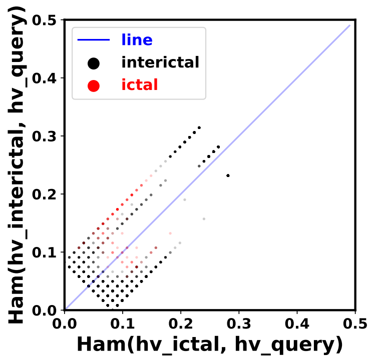
(a) LBP method for Patient_2.



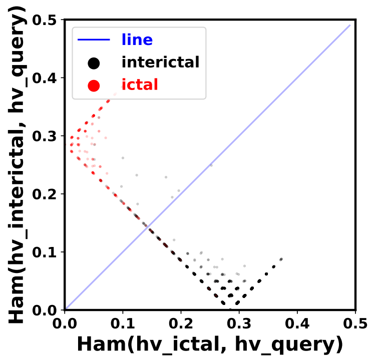
(b) LBP method for Dog_4.



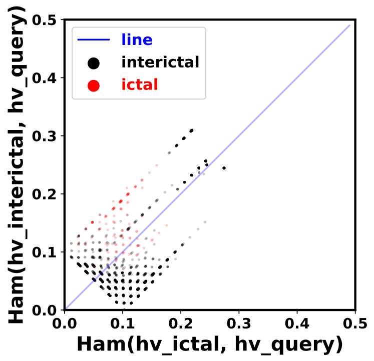
(c) Approach 1 of PSD method using selected features for Patient_2.



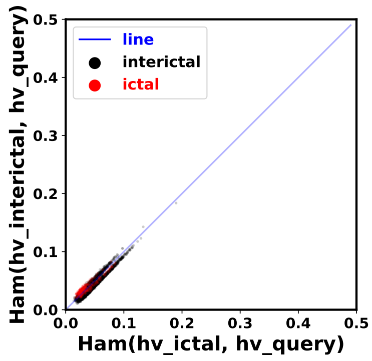
(d) Approach 1 of PSD method using selected features for Dog_4.



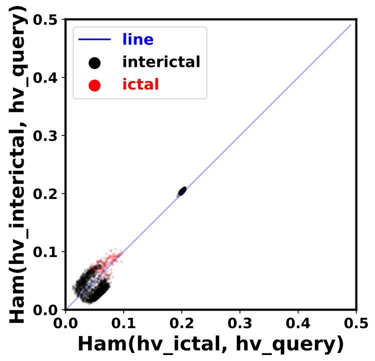
(e) Approach 3 of PSD method using selected features for Patient_2.



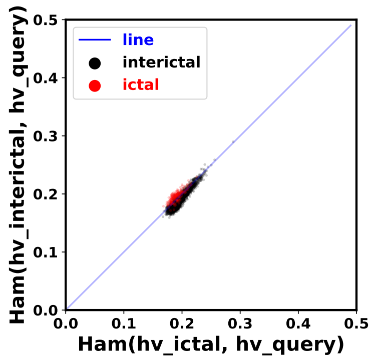
(f) Approach 3 of PSD method using selected features for Dog_4.



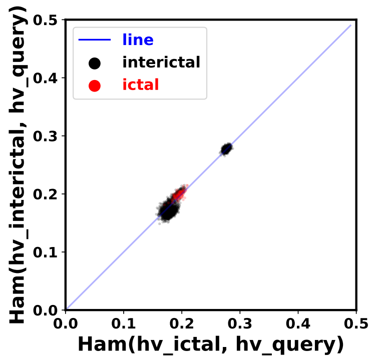
(g) Approach 3 of PSD method using all features for Patient_2.



(h) Approach 3 of PSD method using all features for Dog_4.



(i) Approach 3 of PSD method using all features for Patient_2.



(j) Approach 3 of PSD method using all features for Dog_4.

Figure 3.5: Hypervector distance plots for Patient_2 and Dog_4 among different HD classifiers.

3.5 Conclusion

This research study explores LBP and PSD methods with binary HDC for seizure detection. Note that non-binary HD classification is not presented in this chapter as these do not necessarily outperform binary HD classifiers. We observe that Approach 3 for the PSD method using selected features with Fisher score can achieve the best detection performance on average. By and large, The PSD method outperforms the LBP method in test accuracy, sensitivity, specificity and AUC. Other observations are listed below.

- LBP method utilizes the time-domain information. Ideally, it is suitable for real-time detection. However, the overall performance cannot outperform the PSD methods using the frequency-domain information. From this, we can say that these PSD features are efficient for seizure detection in classifying ictal and interictal data.

- The hypervector distance plot, a powerful visualization tool, should be extended to other binary classification problems in the field of HDC.

- In terms of the memory and complexity requirements, PSD method using selected features outperforms both the LBP method and the PSD method using all features.

Several topics for seizure detection with HDC can be explored further. Future directions may include but are not limited to:

- Binary HDC with more efficient features: Based on this work, PSD features are more efficient than the LBP features. Future work may be directed towards investigating different classes of features that may lead to better performance for this Kaggle dataset with binary HDC. Examples include Mel-frequency cepstral coefficients (MFCCs) [45], wavelets [62, 103], and correlation matrix.

- Feature selection: Inspired from the observation that PSD method using selected features outperforms that using all features, there is a need to develop feature selection or feature ranking approaches for HDC [111]; examples include minimal-redundancy-maximal-relevance (mRMR) [115], MUSE [90], and HD score [56].

- Encoding algorithm: This work mainly uses the record-based encoding [6]. The permutation operation in N-gram-based encoding [7] needs to be investigated in future work.

- Comparison with traditional methods: Although HDC is a viable approach for seizure detection, traditional machine learning approaches can provide better accuracy, sensitivity, and specificity. Whether performance using HD can be further improved to match traditional approaches remains a topic for further investigation. Although the work has described memory and complexity comparisons for different HD approaches, whether the total area and energy consumption of the HD approach can be less than that of traditional approaches needs to be investigated further. A thorough implementation comparison would be useful for understanding area-energy tradeoffs for wearable and implantable devices.

- Group-based seizure detection: The design of group-based (as opposed to patient-specific) seizure detection using HD is also of interest. Then new models do not need to be designed for a new subject.

- Seizure prediction: Finally, the feasibility of HDC using LBP and PSD methods for seizure *prediction* is a topic of further research. Seizure prediction is a harder problem compared to seizure detection.

Chapter 4

Applicability of seizure prediction using HDC

This chapter's content has already been published in [22]. The research aims to answer the question of whether HDC can be applied to seizure prediction. Similar to seizure detection, two encoding approaches are employed: LBP and PSD encoding.

4.1 Introduction

Numerous prior studies have addressed seizure detection (*interictal vs. ictal*) [9, 16, 20, 21, 62, 95, 112] and seizure prediction (interictal *vs. preictal*) [63, 65, 100]. Among these prior studies, the HDC-based local binary pattern (LBP) method, proposed in [16], can achieve 99.36% sensitivity and 95.67% specificity for seizure detection over the SWEC-ETHZ iEEG database [27]. In [112], the applicability of seizure detection using the HDC approach with frequency spectrum features is proven for the CHB-MIT dataset [116]. Additionally, the power spectral density (PSD) method using traditional support vector machines (SVMs) can achieve high performance for both seizure detection and prediction.

Though [21] has studied the applicability of HDC using both LBP and PSD methods for seizure detection, whether these two HDC-based strategies are suitable for seizure prediction has not been investigated. This work applies these two HDC-based encoding approaches for subject-specific seizure prediction using the publicly available Kaggle dataset. The LBP strategy extracts the features from the time domain, whereas the PSD method uses the frequency-domain information. Experimental results indicate that the features generated by the aforementioned two methods are not suitable for an HDC classifier for seizure prediction for this dataset. It has been believed that HDC is applicable to most time-series classification problems. In this chapter, we show that HDC classifiers using the PSD method with a small number of features achieve better performance on the training and validation data as compared to the LBP method.

4.2 Preliminaries

4.2.1 Basics of HDC

In this chapter, the seed hypervectors are either *random* or *level* hypervectors. To put it simply, (a) random hypervectors are quasi-orthogonal to each other and are mainly employed to represent the independently categorical data, e.g., 256 pixel values; (b) level hypervectors are usually linearly correlated and are used to represent the sub-intervals of a given range, e.g., the quantized magnitude of a given time series. The reader is referred to [19, 117] for more details.

4.2.2 Seizure Prediction Dataset

The term “ictal” refers to the period when the subject has a seizure. “Interictal” and “preictal”, respectively, represent the time period at baseline and just before the onset of the seizure. Preictal often refers to the period an hour before the seizure with a 5-minute offset, i.e., the 5-minute period just before the seizure onset is not considered preictal.

The dataset considered in this work is publicly available as part of the Kaggle seizure prediction contest [118]. Such data are provided as 10-min interictal and preictal clips. In total seven subjects are involved: five dogs and two humans. Table 4.1 lists the basic dataset information, including the number of interictal-, preictal-, and test-clips, the number of channels for the iEEG recordings, and the sampling frequency f_s. Each clip is a 10-min iEEG recording. For more detailed data description, interested readers are referred to [99].

Table 4.1: Dataset information.

Subject	#interictal	#preictal	#test	#ch	f_s (Hz)
Dog_1	480	24	502	16	400
Dog_2	500	42	1000	16	400
Dog_3	1440	72	907	16	400
Dog_4	804	97	990	16	400
Dog_5	450	30	191	15	400
Patient_1	50	18	195	15	5000
Patient_2	42	18	150	24	5000

One thing that should be emphasized is that the seizure prediction in this work is a binary classification problem, which identifies the preictal clips among a large number of interictal clips. Ictal clips, which represent the iEEG during the seizure period, are not analyzed. The goal of predicting the preictal clips is to provide sufficient time for warning or prevention before the actual seizure occurs.

4.2.3 Flow Chart of the Employed Approaches

Figure 4.1 shows the flow chart of the employed HDC-based approaches for seizure prediction. For the given multi-channel iEEG recordings, two different types of features are investigated as the input for the HDC classifier: LBP and PSD. To be more specific, the LBP method essentially extracts the time-domain information, whereas the PSD features reflect the frequency-domain information. Note that, there are three types of PSD features: absolute power spectral density (ASP), relative power spectral density (RSP), and the ratio of two ASPs [63]. Finally, after encoding the input features (LBP codes or PSD values), an HDC-based classification is performed for this dataset.

4.2.4 Training and Test Workflow

We conduct the leave-one-seizure-out cross-validation over this seizure prediction dataset. This Kaggle dataset mentioned earlier comes with pre-specified training and test data. Thus the original given training data are separated into the training and validation sets. Note the validation sets are not used for updating the learned HDC model but for evaluating the generalizability of the trained model. To elaborate further, based on

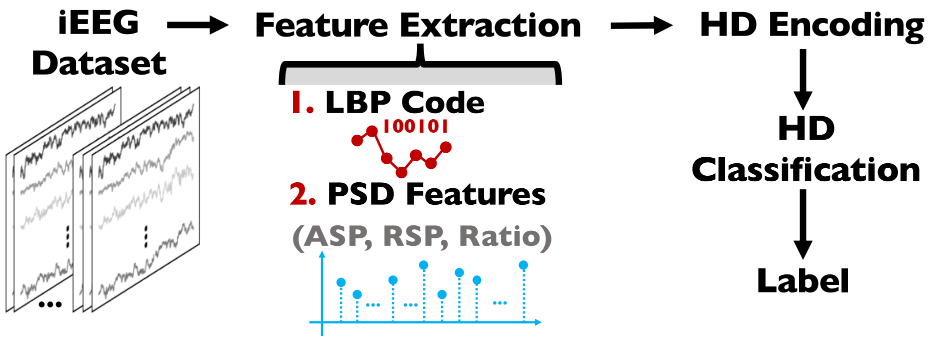


Figure 4.1: Flow chart of the employed approaches.

the number of seizures (S), the original training data are divided into S folds. Within each fold, both the interictal and preictal clips are split into S groups. To be more specific, the preictal clips belonging to the same single seizure form one group, whereas the interictal clips are nearly equally distributed among all S groups. After the data are split, we learn from ($S-1$) groups, validate the trained model on the remaining one group, and test the model over the given test data. The final performance is the average of the results for all S folds.

4.3 Methodology

4.3.1 LBP Method

This work employs the HDC-based LBP method, which is originally proposed in [16] for seizure detection using the SWEC-ETHZ iEEG database [27]. Given the raw iEEG data, LBP codes are extracted as the features, which reflect the time-domain information and are then fed to the HDC classifier.

To extract the LBP codes, consecutive iEEG samples are converted into a bit stream whose components are determined by the sign of the temporal difference of the two adjacent samples. If the difference is positive, then the LBP code is set to be "1"; otherwise it is assigned as "0". Generally, the length of LBP code l should be specified. To obtain a length-l (l-bit) LBP code, $(l+1)$ consecutive time-series data are required. Equations (4.1a)-(4.1e) describe how the class hypervector is generated for either the preictal or interictal class. The parameters, l, N, W, P, K, respectively, represent the length of LBP code, the number of channels, the number of samples within a window, the number of windows, and the number of clips for a single class.

$$\bar{\mathbf{hv}}_{\mathbf{LBP}_{\mathbf{Ch}_{j},i}} \in \{\mathbf{hv}_{\mathbf{LBP}1}, \cdots, \mathbf{hv}_{\mathbf{LBP}2^{l}}\} \tag{4.1a}$$

$$\mathbf{hv}_{\mathbf{spatial}_i} = \Big[\sum\nolimits_{j=1}^{N} \bar{\mathbf{hv}}_{\mathbf{LBP}_{\mathbf{Ch}_{j},i}} \oplus \mathbf{Ch}_j\Big], \tag{4.1b}$$

$$\mathbf{hv}_{\mathbf{win}_p} = \Big[\sum\nolimits_{i=1}^{W-l} \mathbf{hv}_{\mathbf{spatial}_i}\Big], \tag{4.1c}$$

$$\mathbf{hv}_{\mathbf{clip}_k} = \Big[\sum\nolimits_{p=1}^{P} \mathbf{hv}_{\mathbf{win}_p}\Big], \tag{4.1d}$$

$$\mathbf{hv}_{\mathbf{class}} = \Big[\sum\nolimits_{k=1}^{K} \mathbf{hv}_{\mathbf{clip}_k}\Big]. \tag{4.1e}$$

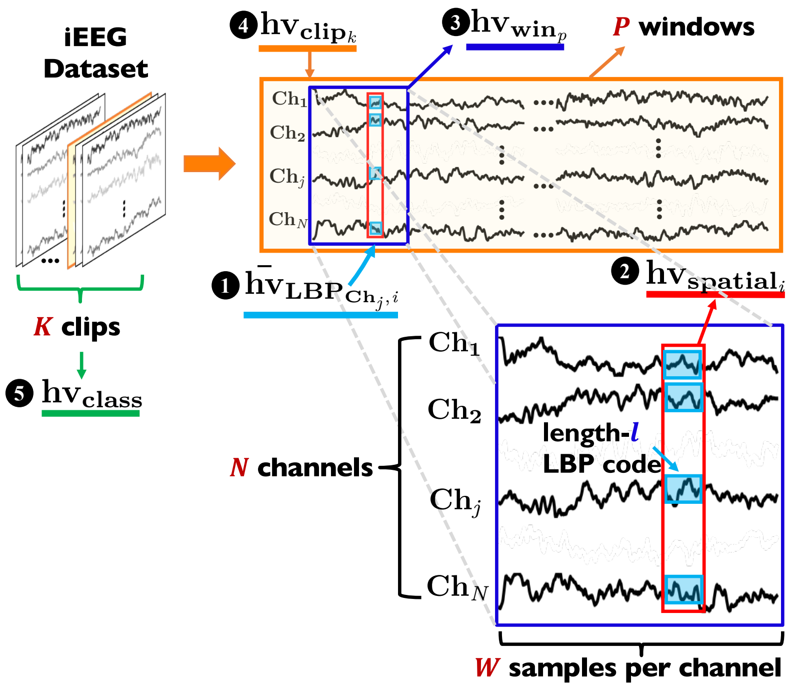

iEEG Dataset
❹ $\mathbf{hv}_{\mathbf{clip}_k}$
❸ $\mathbf{hv}_{\mathbf{win}_p}$
P windows
Ch_1
Ch_2
Ch_j
Ch_N
❶ $\bar{\mathbf{hv}}_{\mathbf{LBP}_{\mathbf{Ch}_{j},i}}$
❷ $\mathbf{hv}_{\mathbf{spatial}_i}$
K clips
❺ $\mathbf{hv}_{\mathbf{class}}$
N channels
length-*l* LBP code
W samples per channel

Figure 4.2: HDC classification using LBP method.

Figure 4.2 illustrates the HDC-based LBP method for seizure prediction. At the very beginning, we generate the seed hypervectors $\{\mathbf{hv}_{\mathbf{LBP}1}, \mathbf{hv}_{\mathbf{LBP}2}, \cdots, \mathbf{hv}_{\mathbf{LBP}2^{l}}\}$ and $\{\mathbf{Ch}_1, \mathbf{Ch}_2, \cdots, \mathbf{Ch}_N\}$ to represent all the 2^l LBP code patterns and N channel indices, respectively. Both of these two sets of hypervectors are random hypervectors. (a) Within each window, the temporal iEEG data are converted into LBP codes $\bar{\mathbf{hv}}_{\mathbf{LBP}_{\mathbf{Ch}_{j},i}}$, where Ch_j specifies the channel information and i indicates the temporal index. These hypervectors are selected from the seed hypervectors according to their code patterns (see Eq. (4.1a)). (b) Computed by Eq. (4.1b), the spatial information across all N channels is encoded by $\mathbf{hv}_{\mathbf{spatial}_i}$, which associates the LBP code patterns with the specific channel. (c) Since there are W samples within a window, the entire window information is represented by $\mathbf{hv}_{\mathbf{win}_p}$, which is calculated by Eq. (4.1c). (d) Afterwards, $\mathbf{hv}_{\mathbf{clip}_k}$ represents a 10-min data as computed in Eq. (4.1d). (e) The final class $\mathbf{hv}_{\mathbf{class}}$ is generated by summing up its constituent clips (as shown in Eq. (4.1e)). For this seizure dataset, we use a 1s-window for each 10-min clip.

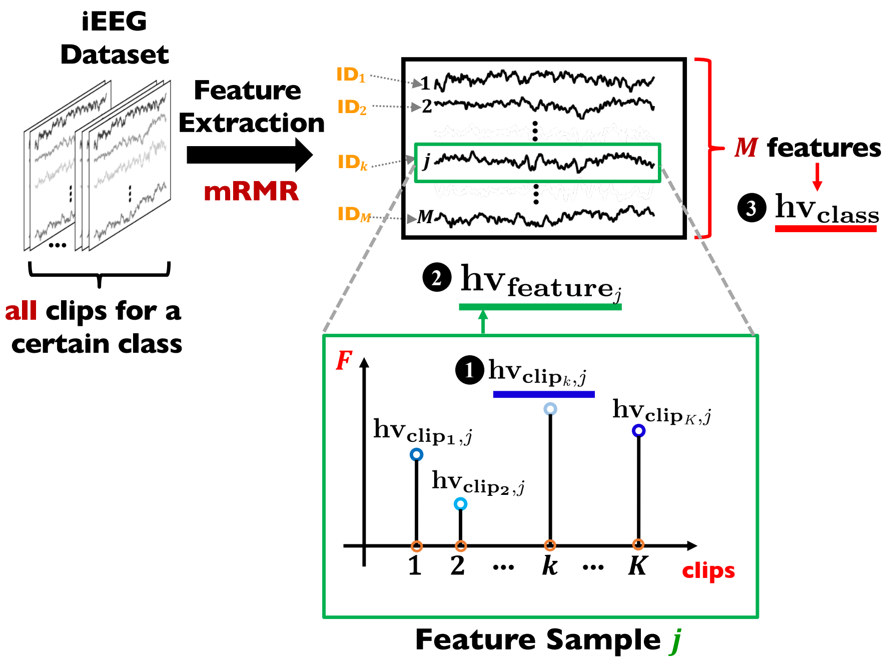


Figure 4.3: HDC-based PSD method for seizure prediction.

4.3.2 PSD Method

PSD features manifest the frequency-domain information. Unlike [65], where only RSP and ratio of spectral powers are considered, this research employs one more type of PSD feature—ASP. How these three PSD features are computed is described in [64] (see page 2).

We follow the same sub-band split for the PSD feature extraction as described in [65], whose classifier is polynomial SVM. (a) For dogs, the frequency band is divided into 10 sub-bands (Hz): 3-8, 8-13, 13-30, 30-55, 55-80, 80-105, 105-130, 130-150, 150-170, 170-200. (b) For human patients, two more sub-bands are included: 200-225 and 225-250. Note that the power line noise at 60 Hz and its harmonics should be eliminated. The reader is referred to Sec.II.C of [65] for more details. In this work, we have $65(= 10 + 10 + \binom{10}{2})$ PSD features for dogs and $90(= 12 + 12 + \binom{12}{2})$ for human patients. Equations (4.2a)-(4.2c) describe how the class hypervector is generated for the PSD method. The parameters, q, K, M, respectively, represent the quantization level, the number of clips for a single class, and the number of selected features.

$$\mathbf{hv}_{\mathbf{clip}_{k},j} \in \{\mathbf{L_1}, \mathbf{L_2}, \cdots, \mathbf{L}_q\}, \text{ where } k \in [1, K], \tag{4.2a}$$

$$\mathbf{hv}_{\mathbf{feature}_j} = \left[\sum\nolimits_{k=1}^{K} \mathbf{hv}_{\mathbf{clip}_{k},j}\right], \tag{4.2b}$$

$$\mathbf{hv}_{\mathbf{class}} = \left[\sum\nolimits_{j=1}^{M} \mathbf{hv}_{\mathbf{feature}_j} \oplus \mathbf{ID}_j\right]. \tag{4.2c}$$

We employ the record-based encoding (summarized in [19] and is the “Approach 3” in [21]) using a small number of features that are selected by minimum redundancy maximum relevance (mRMR). After calculating PSD values for each feature, we concatenate the corresponding interictal and preictal categories of the training data together and normalize them into the range of [0,1]. Then use the training information to scale both the validation and test data. Figure 4.3 shows the HDC-based classification for seizure prediction using PSD features. First, seed hypervectors are generated $\{\mathbf{ID}_1, \mathbf{ID}_2, \cdots, \mathbf{ID}_M\}$ and $\{\mathbf{L}_1, \mathbf{L}_2, \cdots, \mathbf{L}_q\}$ to represent the feature identifiers and the quantized PSD values, respectively. Note that the feature identifier hypervectors are random hypervectors, whereas the quantization hypervectors are level hypervectors. The class hypervectors are generated following these steps: (a) According to the quantized PSD values, hypervectors $\mathbf{hv}_{\mathbf{clip}_{k},j}$ are selected from the seed hypervectors, where “clip_k, j” specifies that this PSD value is calculated from the clip_k to form the feature j (as shown in Eq. (4.2a)). (b) As described in Eq. (4.2b), for the feature j, the corresponding $\mathbf{hv}_{\mathbf{feature}_j}$ hypervector is obtained by adding all its constituent K clip hypervectors. (c) The class hypervector $\mathbf{hv}_{\mathbf{class}}$ is generated as shown in Eq. (4.2c). (d) Similar to [65], we use a 2s-window with 50% overlap to generate the PSD feature

Table 4.2: Experimental results for LBP method with the code length $l = 6$.

Patient	**Training Data**				**Validation Data**				**Test Data**			
	ACC	**Sen.**	**Spec.**	**AUC**	**ACC**	**Sen.**	**Spec.**	**AUC**	**ACC**	**Sen.**	**Spec.**	**AUC**
Dog_1	67.33	59.72	67.71	0.64	64.29	8.33	67.08	0.38	58.72	39.58	59.68	0.50
Dog_2	70.13	67.46	70.36	0.69	69.02	66.67	69.22	0.68	58.41	40.63	60.17	0.50
Dog_3	66.41	69.19	66.27	0.68	63.43	54.17	63.89	0.59	63.51	23.41	65.45	0.44
Dog_4	69.84	74.04	69.33	0.72	68.76	73.53	68.34	0.71	68.50	67.39	68.56	0.68
Dog_5	68.07	69.17	68.00	0.69	67.08	66.67	67.11	0.67	37.70	8.33	39.66	0.24
Patient_1	63.64	77.78	58.33	0.68	27.27	11.11	33.33	0.22	44.27	80.56	41.89	0.61
Patient_2	77.50	66.67	82.14	0.74	56.67	38.89	64.29	0.52	47.78	38.10	48.77	0.43
mean	68.99	69.15	68.88	0.69	59.50	45.62	61.89	0.54	54.13	42.57	54.89	0.49

values for each 10-min clip.

4.4 Experimental Results

Using the aforementioned LBP and PSD methods, the corresponding experimental results are reported in Tables 4.2 and 4.3, respectively. We can observe from these two tables that: (a) the LBP method achieves the on average 0.69 training AUC, 0.54 validation AUC, and 0.49 test AUC (Table 4.2), whereas the PSD method leads to 0.75 training AUC, 0.62 validation AUC, and 0.51 test AUC (Table 4.3). Therefore, the PSD method performs better than the LBP method for seizure prediction on the training and validation data. However, both methods perform no better than a random guess for the test data. (b) The HDC-based LBP method achieves 99.36% sensitivity and 95.67% specificity for seizure detection in [16]. However, this method does not perform well in seizure prediction. The LBP method could be suitable for Dog_4 since it achieves 0.72, 0.71, and 0.68 AUC for training, validation, and test data. The best result using the LBP method is achieved for Dog_4. (c) The PSD method can achieve 0.98 validation AUC in [65] using polynomial SVM. However, only 0.64 validation AUC is achieved by the HDC-based PSD method in this work. Note that, the HDC-based PSD method has been investigated in [21] to demonstrate that HDC is suitable for seizure detection. However, the PSD features cannot always predict seizures using HDC for the test data in this dataset.

The results of the top ten competitors from the Kaggle seizure prediction contest have been summarized in [99], where the test AUC scores range from 0.82 to 0.86 (see

Table 4.3: Experimental results for PSD method with the quantization level $q = 21$.

Patient	**Training Data**				**Validation Data**				**Test Data**			
	ACC	**Sen.**	**Spec.**	**AUC**	**ACC**	**Sen.**	**Spec.**	**AUC**	**ACC**	**Sen.**	**Spec.**	**AUC**
Dog_1	91.53	33.33	94.44	0.64	90.48	0.00	95.00	0.48	87.00	11.46	90.79	0.51
Dog_2	89.39	64.68	91.48	0.78	88.13	47.62	91.55	0.70	65.83	33.02	69.07	0.51
Dog_3	95.01	34.85	98.02	0.66	94.51	30.56	97.71	0.64	81.25	16.67	84.38	0.51
Dog_4	86.83	34.84	93.14	0.64	85.96	26.08	93.24	0.60	93.04	8.77	98.18	0.53
Dog_5	95.94	65.83	97.94	0.82	95.83	66.67	97.78	0.82	93.72	0.00	100.00	0.50
Patient_1	92.42	72.22	100.00	0.86	83.33	44.44	97.92	0.71	63.59	44.44	64.85	0.55
Patient_2	90.00	66.67	100.00	0.83	51.67	16.67	66.67	0.42	62.22	19.05	66.67	0.43
mean	91.59	53.20	96.43	0.75	84.27	33.15	91.41	0.62	78.09	19.06	81.99	0.51

Table 2 in [99]). None of them is based on HDC, which can be beneficial for lightweight classifiers and wearable devices.

4.5 Conclusions

The HDC PSD method works well for the two human subjects and two dogs for the training data. Generally, the PSD method achieves better performance than LBP using HDC; however, neither of them is practically applicable for seizure prediction by testing over the Kaggle dataset. Significant portions of the test samples in the Kaggle dataset are from out-of-distribution data as reported in [99]. Predicting seizures is a hard problem due to the highly non-stationary nature of brain signals. Thus, further research needs to be directed towards the development of HDC approaches that are robust to classifying out-of-distribution data. Although the LBP approach exploits some aspects of temporal properties, the PSD method does not. Future work should, therefore, be directed towards exploiting temporal properties in the context of PSD features. In addition, new encoding approaches that can take advantage of temporal properties in the context of HDC classifiers could be explored. Features could also be generated by neural networks such as autoencoders.

Chapter 5

Classifying functional brain graphs using graph hypervector representation

The work for this chapter was published in [23]. Theoretically, the brain network can be denoted by a functional brain graph, which consists of brain regions/nodes and edges. Previous studies have shown that brain graph classification can be effectively tackled by utilizing features extracted from the graph structure [119]. These features specifically refer to subgraph entropies. This chapter explores the potential benefits of utilizing a graph representation called GrapHD, which is based on HDC, for the classification of brain graphs, instead of relying on those more complex features.

5.1 Introduction

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 [119]. Researchers have a great interest in understanding brain connectivity patterns through the lens of network theory. The corresponding hidden assumptions are (a) 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. (b) Although the brain states change dynamically when tasks or no-tasks are conducted, each state has its specific pattern, namely the brain graph. (c) Network metric is group differentiating and biologically meaningful. Built on these three assumptions, [119] 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 [120].

Most HDC applications deploy two encoding approaches: record-based encoding and N-gram-based encoding [19]. Both of these have been applied to efficiently represent the data structure like "sequence" [16, 19, 56]. A novel encoding approach, called GrapHD, has been recently proposed to encode the graph structure [13] as a graph hypervector. In [13], 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 work addresses the group-based classification of graphs using 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 subgraphs 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 work 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 [13]. In this chapter, we extend the applicability of GrapHD for classification. Third, as compared to fully connected brain graphs, sub-graphs can improve the performance of HDC classifiers.

5.2 Methodology

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

5.2.1 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 [121]. 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 [21]) or the *edge weight* hypervectors $\mathbf{E}_w$ (Sec. 5.2.2), whereas the latter ones are random-hypervectors. Then record-based encoding is applied to this data structure as follows: (a) 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 Eq. (5.1a), 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. (b) The compound hypervector $\mathbf{S}_i$ for this vector can be calculated by Eq. (5.1b). (c) The class hypervector $\mathbf{hv_{class}}$ is formed by adding its constituent hypervectors $\mathbf{S}_i$ as described in Eq. (5.1c).

$$\bar{\mathbf{V}}_k \in \{\mathbf{L_1}, \cdots, \mathbf{L_q}\}, \quad (\text{ or } \bar{\mathbf{V}}_k = \mathbf{E}_w), \tag{5.1a}$$

$$\mathbf{S}_i = \bar{\mathbf{V}}_1 * \mathrm{ID}_1 + \cdots + \bar{\mathbf{V}}_k * \mathrm{ID}_k + \cdots + \bar{\mathbf{V}}_{\binom{m}{2}} * \mathrm{ID}_{\binom{m}{2}}, \tag{5.1b}$$

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

$$\mathbf{hv_{class}} = \textstyle\sum_{i=1}^{N} \mathbf{S}_i, \text{ where } N \text{ is the number of subjects.} \tag{5.1c}$$

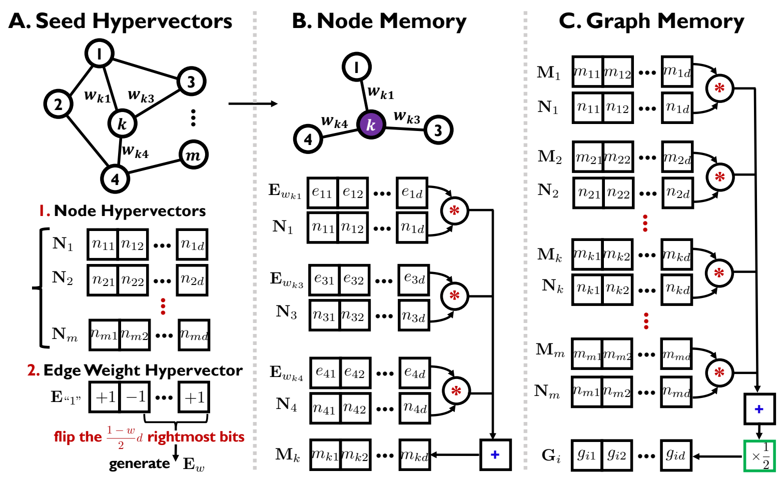


Figure 5.1: Graph encoding for weighted undirected graphs in GraphHD [13].

5.2.2 GraphHD Encoding

GraphHD is proposed to represent an arbitrary graph which is composed of nodes and edges [13]. Such an encoding approach can represent: (a) unweighted undirected, (b) unweighted directed, (c) weighted undirected, and (d) 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.

Figure 5.1 illustrates how to encode an arbitrary weighted undirected graph with m nodes using GraphHD. (a) 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 $(\mathbf{E}_{\text{“}1\text{”}})$ 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 $\mathbf{E}_{\text{“}1\text{”}}$. This flipping operation ensures that the cosine similarity between w and the 1 vector, $\mathbf{E}_{\text{“}1\text{”}}$, is w (see Eq. (5.3)). (b) The node memory $\mathbf{M}_k$ reflects the edge information regarding the node k as shown in Eq. (5.2b). (c) The whole graph is encoded as described in Eq. (5.2c). Note that the query hypervectors can be generated from these three steps Eqs. (5.2a-5.2c), whereas class hypervectors are obtained by Eq. (5.2).

$$\mathbf{N}_k \in \{\mathbf{N}_1, \mathbf{N}_2, \cdots, \mathbf{N}_m\},\ \text{weight value "1": } \mathbf{E}_{\text{"1"}} \tag{5.2a}$$

$$\mathbf{M}_k = \textstyle\sum_k \mathbf{E}_{w_{kj}} * \mathbf{N}_j,\ \text{where } j \in \{\text{nodes adjacent to node } k\}, \tag{5.2b}$$

$$\mathbf{G}_i = \frac{1}{2}\sum_k \mathbf{M}_k * \mathbf{N}_k,\ \text{where } 1 \leq k \leq m, \tag{5.2c}$$

$$\mathbf{hv_{class}} = \textstyle\sum_{i=1}^{N} \mathbf{G}_i,\ \text{where } N \text{ is the number of subjects.} \tag{5.2d}$$

$$\delta(\mathbf{E}_w, \mathbf{E}_{\text{"1"}}) = \frac{\mathbf{E}_w \cdot \mathbf{E}_{\text{"1"}}}{d} = \frac{\frac{1+w}{2}d - \overbrace{\frac{1-w}{2}}^{\text{flip these bits}}d}{d} = w \tag{5.3}$$

5.3 Materials

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 work is also included.

5.3.1 HCP Dataset and Preprocessing

In this work, we use the publicly available dataset from the HCP [119], where seven different task-fMRI data are collected from 475 healthy subjects. Similar to [119], two chosen tasks are investigated in this work: 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 [119] 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 [119], this work essentially addresses the brain graph classification from fMRI data at a group level, where only 85 RoIs are considered. However, unlike [119] where entropy measures are used as features, this research considers a graph where the edge weight corresponds to the absolute value of the correlation coefficient.

5.3.2 Flowchart of The Approaches

Figure 5.2 illustrates how the two HDC encoding approaches are applied for brain graph classification using the fMRI data in the HCP study. Following the setup of [119], preprocessing 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 [119]). 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. GraphHD is adopted once the graph structure is built. HDC classification is carried out after the information is efficiently encoded.

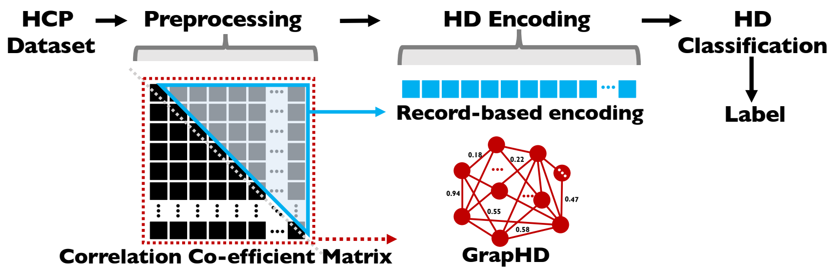


Figure 5.2: Flowchart of the two HDC approaches for the brain state classification.

5.4 Experimental Results

5.4.1 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).

Table 5.1: Classification performance for three classification tasks.

Approach	**Emotion *vs.* Gambling**				**Emotion *vs.* No-task**				**Gambling *vs.* No-task**			
Metrics[1]	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

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

5.4.2 Evaluation Metrics

For this classification problem, following a similar evaluation setting in [119], four metrics are measured for the performance evaluation: accuracy, sensitivity, specificity, and area under the ROC curve (AUC), where ROC refers to the receiver operating characteristic.

5.4.3 Performance Comparison

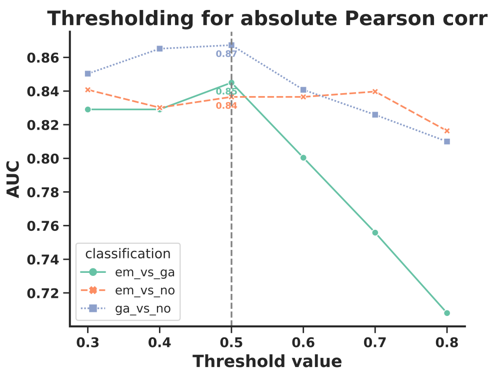
Table 5.1 demonstrates the simulation results for the two employed HDC encoding approaches given the same features—85×85 correlation-coefficient matrix. Additionally, the results of the traditional linear SVM approach are also considered as a reference. From this table, we observe that: (a) with the same correlation-coefficient matrices as the features, the GrapHD achieves similar performance as the record-based encoding. (b) 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 5.2, 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.

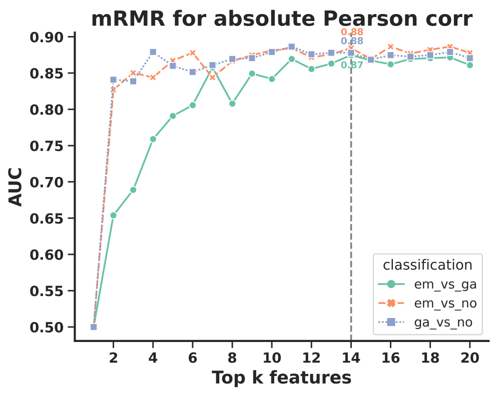
Table 5.2: Memory storage[1] comparison for two HD encoding approaches.

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

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



(a) Threshold strategy.



(b) mRMR algorithm.

Figure 5.3: Performance improvement for GrapHD encoding by two strategies.

5.4.4 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.

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 Figure 5.3(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.

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 $k \leq 20$. As shown in Figure 5.3(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.

5.5 Conclusion

This chapter 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 chapter 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.

Chapter 6

Clustering using HDC

The work for this chapter has been submitted to [24]. The existing HDC-based clustering algorithm, HDCluster [14], can be significantly influenced by the random seed used to generate seed hypervectors, resulting in high variation in clustering performance. To address this limitation, we propose more robust HDC-based clustering algorithms in this chapter.

6.1 Introduction

Clustering is one of the fundamental tasks in machine learning that seeks to create clusters/groups of similar data. Traditional clustering methods, such as k-means [122, 123] and hierarchical clustering [124–126], can have difficulty when addressing the high-dimensional data due to the "curse of dimensionality": distance measures become increasingly meaningless as the number of the dimensions of a dataset increases [127]. Due to the nature of data representation in HDC, the original data point is represented in hyperdimensional space as a single hypervector, regardless of its original dimension. Therefore, HDC is a promising alternative for clustering high-dimensional data.

For clustering using HDC, [128] explains both analytically and empirically why the random mapping in HDC data representation can approximately preserve the mutual similarity among the original data. As an alternative to dimensionality reduction, random projection has been demonstrated to achieve comparable data separability with negligible computational complexity with principal component analysis (PCA) for document classification. Therefore, as indicated by [128], HDC is promising for faster clustering, especially in situations when the original data has a very large dimensionality. An approach, referred to as HDCluster, was presented in [14] to systematically cluster data in the HDC domain. This work showed that HDCluster can be more accurate than traditional k-means over diverse datasets. An in-storage computing solution, called Store-n-Learn, is proposed for HDC-based classification and clustering across the flash hierarchy in [129]. Experiments show that Store-n-Learn can achieve on average 543× faster than CPU for clustering over ten datasets. In [130], a processing-in-memory (PIM) architecture that utilizes HDC for a more robust and efficient machine learning system was proposed. In particular, clustering is supported by HyDREA to achieve 32× speed up and 289× energy efficiency than the baseline architecture. It may be noted that the HDC-based clustering in [14, 128] is algorithm-oriented, whereas [129, 130] emphasize the hardware implementations using the same clustering framework as [14].

One drawback of the HDCluster is that the assignment of initial cluster hypervectors using random hypervectors can lead to a high variance in clustering performance when random hypervectors are generated based on different seeds. This necessitates the design of new clustering algorithms in the HDC domain that are robust. In this work, we propose four novel HDC-based clustering algorithms that can learn from the encoded data. These encoded data are referred to as *query* hypervectors.

Leaked from the encoded query hypervectors, intra-cluster hypervectors are much closer/similar than inter-cluster hypervectors. Therefore, the categorized information can be inferred by the similarity results among these query hypervectors. In this chapter, four HDC-based clustering algorithms are proposed: similarity-based k-means, equal bin-width histogram, equal bin-height histogram, and similarity-based affinity propagation. The first three algorithms require only one-dimensional similarity results, while the fourth algorithm requires a matrix of similarity results. In this work, the emphasis is more on the algorithms than on the hardware implementation.

The contribution of this work can be summarized as follows: (a) Due to the use of random projections, HDC is believed to have robust performance. It should be noted, however, that different initializations of random hypervectors may result in high variance. Using different seeds of the random number generator, we find that HDCluster's clustering performance is not as robust as expected. Notably, this observation entails the initialization of random hypervectors, which has been neglected in previous HDC studies. (b) In contrast to HDCluster's random assignment for the initial cluster hypervectors in the data space, we learn from the data by leveraging the fact that intra-cluster hypervectors have a higher similarity than inter-cluster hypervectors. To achieve this, the similarity results among query hypervectors are utilized. Our proposed HDC-based clustering algorithms are more robust in clustering performance. In addition, our algorithms achieve higher clustering accuracy, fewer iterations for updating cluster hypervectors, and less program execution time as compared to the existing HDCluster. Particularly, similarity-based affinity propagation always shows higher accuracy than the traditional k-means, hierarchical clustering, and other HDC-based clustering algorithms over all tested eight datasets. (c) The effectiveness of the projection onto hyperdimensional space on clustering performance is examined by applying three traditional clustering algorithms to both the original data and the encoded data. According to the experimental results over eight datasets, five out of eight can achieve higher or comparable clustering accuracy. Additionally, these results imply that maintaining the original space is preferable if the number of clusters is large, e.g., $k = 26$.

6.1.1 Traditional Clustering Algorithms

Traditional K-means

To create k clusters from the given N data points, as a simple and efficient algorithm, the traditional k-means is widely used to find the local optimal solution [122, 123]. Its goal is to minimize the sum of the squared distances (denoted as ϕ) between every data point and its associated cluster center. In k-means, the initial k clusters are randomly selected from the data domain. ❶ Each data point is then assigned to the closest cluster center. ❷ After all data points are assigned, each cluster center is recomputed/updated by the mean of its constituent data points. Repeat ❶-❷ until ϕ converges or this algorithm exceeds the pre-defined maximum iteration number.

Traditional Hierarchical Clustering

This algorithm aims to build the hierarchy of clusters so that the clusters are organized in a tree-like structure [124–126]. There are two typical methods to conduct hierarchical clustering: bottom-up and top-down. In general, the top-down method is more computationally expensive and may not always produce well-defined clusters. Thus in this paper, we use bottom-up hierarchical clustering. To put it simply, bottom-up hierarchical clustering starts with an $N \times N$ matrix of pairwise distances that are computed from the given N data samples. Initially, each data sample is viewed as a unique cluster. Therefore, there are N clusters at the very beginning. ❶ Pairs of clusters that have the closest distance are merged as a new cluster so that this new cluster is computed as the average of all the data points that belong to the merged clusters. ❷ The size of the matrix of distances is reduced by 1 and this distance matrix should be recalculated. Repeat ❶-❷ until all the data points are merged into one cluster.

Traditional Affinity Propagation

This algorithm identifies a subset of representative examples (called "*exemplars*") from the clustered data by passing messages between the data points. It takes the $N \times N$ similarity matrix between all data points as an input. Two kinds of messages are exchanged between data points: *responsibility* and *availability*. The responsibility ($r(i, k)$) reflects how well-suited the point k is to serve as the exemplar for point i, whereas the availability ($a(i, k)$) represents how appropriate the point i chooses the point k as its exemplar. Both responsibility and availability are iteratively updated based on the messages passed between data points until this algorithm converges. Finally, each data point is assigned to a cluster based on its exemplar. Interested readers are referred to [131] for more details.

6.1.2 HDCluster

HDCluster [14] is a framework for HDC-based clustering, which is inspired by the traditional k-means clustering algorithm. Figure 6.1 illustrates an overview of HDCluster. For a k-cluster problem, the dataset contains N data samples with n features. At the very beginning, the initial k cluster centers are represented by k random hypervectors.

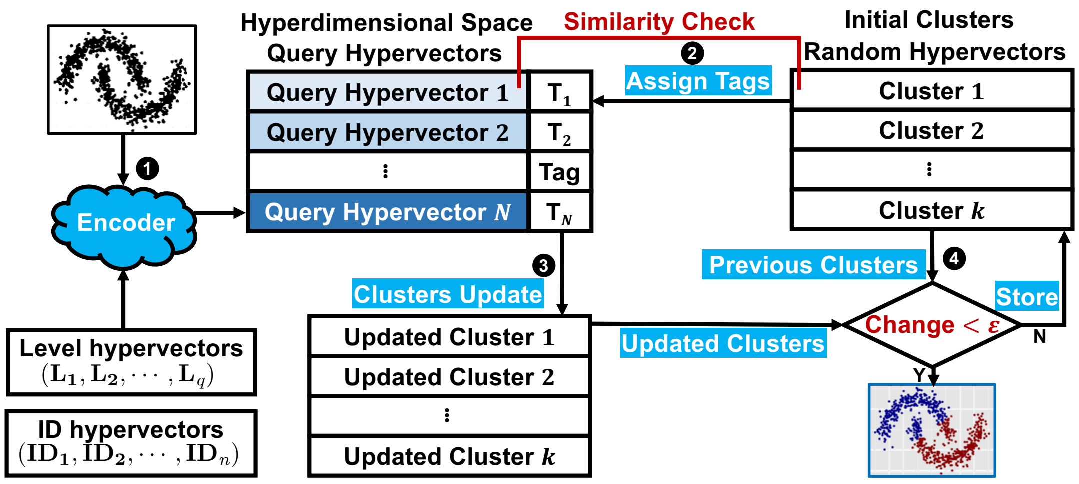


Figure 6.1: HDCluster overview [14]. In [14], the encoder of the original HDCluster refers to record-based encoding, and the initial cluster centers are random hypervectors.

❶ After the feature values are quantized into q levels, each data sample is encoded by the record-based encoding, where the feature indices are represented by **ID** hypervectors (random hypervectors), and the feature values are encoded by $\mathbf{V} \in \{\mathbf{L}_1, \cdots, \mathbf{L}_q\}$ (level hypervectors). ❷ In each iteration, each data sample is assigned to its cluster center which reflects the highest similarity and is assigned a tag to represent the corresponding cluster. ❸ The cluster hypervectors are updated/regenerated by adding their associated data samples. ❹ The iterations will be terminated if (i) there is a minor change for the center hypervectors between two consecutive iterations or (ii) it exceeds the pre-defined number of iterations.

In the traditional k-means, the initial cluster centers are assigned from the data domain [123]. However, the assignment of cluster centers in HDCluster is from the data space ($\{0, 1\}^d$). This assignment of initial cluster hypervectors in the HDCluster leads to non-robust clustering performance. More details are discussed in Sec. 6.4.

6.2 Methodology

In this section, we propose four novel HDC-based clustering algorithms. Different from HDCluster, the assignment of all these algorithms for k initial center hypervectors is from the data domain. To be more specific, the initial center hypervectors in this work are no longer random hypervectors.

In HDC, original data points are encoded as hypervectors; thus, they are projected onto the hyperdimensional space. The relationships among those hypervectors are determined by their similarity measurement. Therefore, our HDC-based clustering algorithms are proposed based on the following assumptions: (i) the projection onto hyperdimensional space is helpful for data separability; (ii) the similarity measurement among hypervectors leaks the information for clustering/grouping the data.

To cluster the given N data samples with our proposed HDC-based clustering algorithms below, the first three algorithms only require computing the similarity measurement $(N-1)$ times if we randomly pick a data sample as the starting point. However, the fourth algorithm needs to compute $\binom{N}{2}$ similarity measurements.

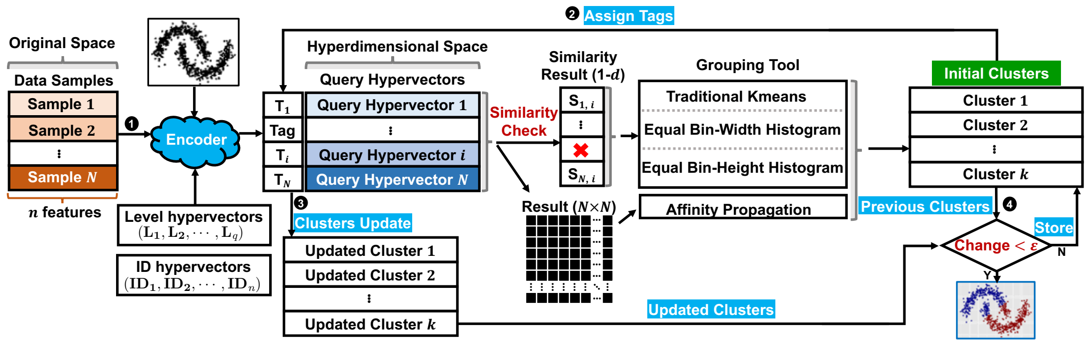


Figure 6.2: Proposed HDC-based clustering algorithms.

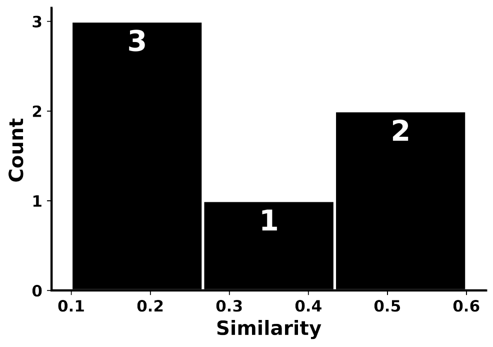
6.2.1 Similarity-Based K-means

As shown in Figure 6.2, after the original data samples are encoded as query hypervectors, a starting point is randomly chosen among all N data samples. Conduct the similarity measurement over all the other $(N-1)$ data samples with this starting point, a 1-d similarity result is obtained. A traditional k-means algorithm is then applied to this 1-d similarity result to obtain the k clusters.

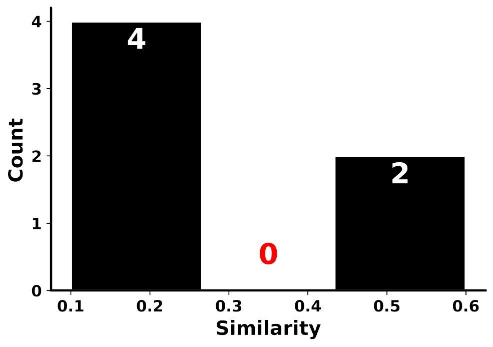
6.2.2 Equal Bin-Width Histogram

As a histogram is used to reflect the distribution of 1-d data, it is natural to use a histogram to cluster or group data. The histogram is designed to have k bins for any

k clustering problem. There are two typical styles of histograms: equal bin width and equal bin height. In this case, cluster hypervectors are evenly distributed in the hyperdimensional space between the data samples if a bin-width histogram is applied to the calculated 1-d similarity result. Figure 6.3 shows two toy examples of six query hypervectors to be allocated into three clusters using the equal bin-width histogram.



(a) Histogram with no zero-membered bins.



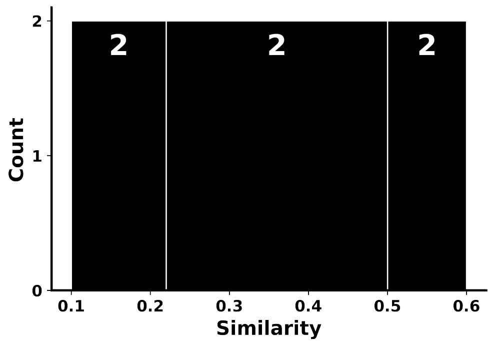
(b) Histogram with zero-membered bins.

Figure 6.3: Examples for the equal bin-width histogram.

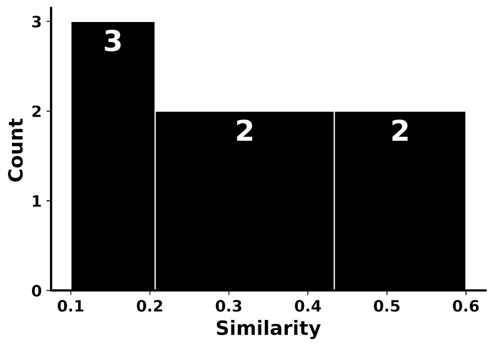
However, such a uniform distribution of cluster hypervectors cannot be guaranteed for any clustering problem. The histogram will contain some zero-membered bins in practice, e.g., Figure 6.3(b). These zero-membered bins can result in non-convergence for the update of cluster center hypervectors. To solve this issue, this algorithm is considered unsuitable for clustering the input data.

6.2.3 Equal Bin-Height Histogram

If an equal-height histogram is employed to the 1-d similarity result, the corresponding assumption is that all these k clusters have similar group sizes, i.e., the data are sampled from a uniform probability distribution. This equal bin-height histogram method is preferred when prior knowledge that the clusters have similar member sizes is given. Figure 6.4 gives two toy examples for the equal bin-height histogram. For the number of compared query hypervectors that can be divisible by the number of k clusters ($\text{Mod}(\frac{N-1}{k}) = 0$, where Mod represents reminder), each cluster has the exact same group size, whereas the group sizes of k clusters are roughly similar for the indivisible case ($\text{Mod}(\frac{N-1}{k}) \neq 0$).



(a) Divisible case.



(b) Indivisible case.

Figure 6.4: Examples for the equal bin-height histogram.

6.2.4 Similarity-Based Affinity Propagation

Different from the previous three algorithms, this algorithm requires an $N \times N$ similarity result. Then this similarity result is fed into the traditional affinity propagation algorithm [131] to obtain the clustering results.

Different from traditional affinity propagation whose similarity matrix is typically measured by the pairwise Euclidean distance of the data points in the original space, our proposed similarity-based affinity propagation starts with the similarity result measured by Eq. (2.3) for the encoded data (query hypervectors) in the hyperdimensional space.

6.3 Materials

6.3.1 Dataset Description

In this paper, we apply our HDC-based clustering algorithms to eight diverse datasets that are also tested in [14]. The corresponding ground truth is provided. More details are described in Table 6.1.

Table 6.1: Datasets for clustering using HDC.

Datasets	**MNIST**	**ISOLET**	**IRIS**	**Glass**	**RNA-seq**	**Cancer**	**Ecoli**	**Parkinson's**
# of Data Samples (N)	10,000	7,797	150	214	801	569	336	195
# of Features (n)	784	617	4	9	20,531	30	7	22
# of Clusters (k)	10	26	3	6	5	2	8	2

6.4 Experimental Results

6.4.1 Experimental Setup

We implement both HDCluster and our proposed HDC-based clustering algorithms using Python implementation. For hypervectors generation, we employ the library "torchhd" [117]. We evaluate the clustering algorithms by three metrics: accuracy (ground truth is known), number of iterations for the convergence of cluster hypervectors, and execution time. As mentioned above, the generation of random hypervectors has an impact on the clustering performance of HDCluster. To capture the non-robust performance, we test all the HDC-based clustering algorithms over 500 runs with different pseudo seeds. Thus in Python code, the "torch.manual_seed" ranges from 0 to 499. This enables us to compute the variance of the performance results. Note that, in HDCluster, the algorithm was run only once and no variance was reported.

The projection onto hyperdimensional space requires the quantization of the original data. As with [14], quantization level q is set to 16 to ensure a fair comparison. Additionally, the dimensionality of hypervectors is set as $d = 10,000$ for all experiments.

To measure the *minor* change of cluster hypervectors between two consecutive iterations, the minimum cosine similarity (over k) between current ($\{\mathbf{C_1}', \cdots, \mathbf{C_k}'\}$) and previous ($\{\mathbf{C_1}, \cdots, \mathbf{C_k}\}$) cluster hypervectors should be greater than 0.99 for non-binary HDC, whereas for the maximum Hamming distance between current and previous cluster hypervectors should be less than 0.01 for binary HDC. The maximum number of iterations for termination of the iterative update is predefined as 300 in this work.

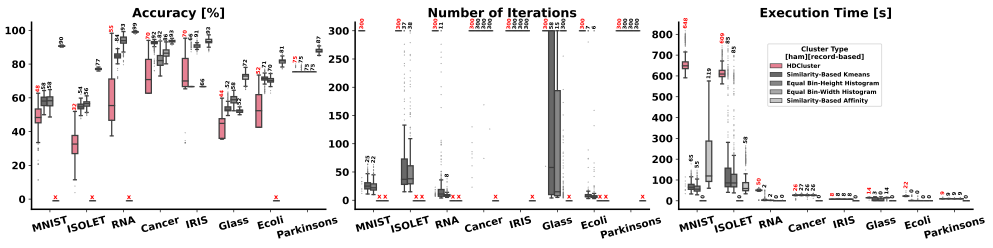
Note in [14], both the seed hypervectors and the compound hypervectors are binary hypervectors ($\in \{0, 1\}^d$). In this chapter, similar to [14], we test our proposed HDC-based algorithms using binary hypervectors. We also apply our algorithms using bipolar seed hypervectors and integer compound hypervectors.

6.4.2 Comparison with HDCluster

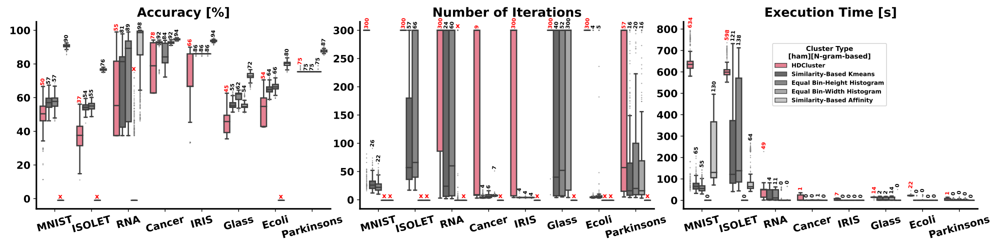
Using HDCluster as a baseline framework, we test our proposed HDC-based clustering algorithms over eight datasets. Two encoding algorithms, record-based and N-gram-based encoding, are employed. Both binary ($\{0, 1\}^d$) and bipolar ($\{-1, 1\}^d$) seed hypervectors are examined. Additionally, we are also interested in one-pass clustering in addition to the iterative update of cluster hypervectors.

Iterative Update of the Center Hypervectors

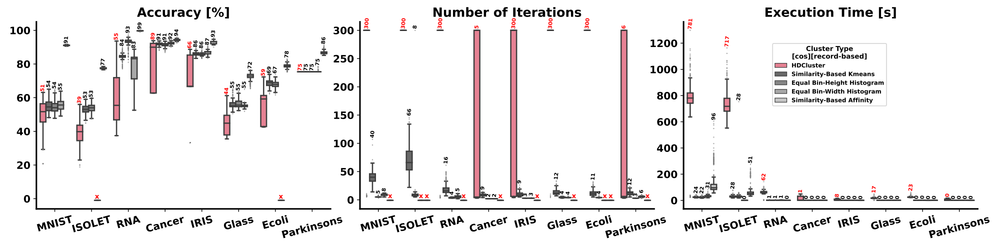
Figure 6.5 shows the experimental results of our proposed HDC-based clustering algorithms and the baseline HDCluster over 500 runs using boxplots. Median values are annotated on the top of the boxplots. As shown in Figure 6.5, our proposed algorithms are more robust in clustering accuracy, require fewer iterative updates in cluster hypervectors, and consume less execution time over all eight datasets as compared to HDCluster. Among our proposed HDC-based clustering algorithms, similarity-based affinity propagation always achieves the highest clustering accuracy over all eight datasets. To elaborate a little further, clustering accuracy is significantly improved by this similarity-based affinity propagation for MNIST ($\approx$38%), ISOLET ($\approx$24%), Glass ($\approx$17%), Ecoli ($\approx$9%), and Parkinson's ($\approx$9%), as compared to the other HDC-based clustering algorithms.



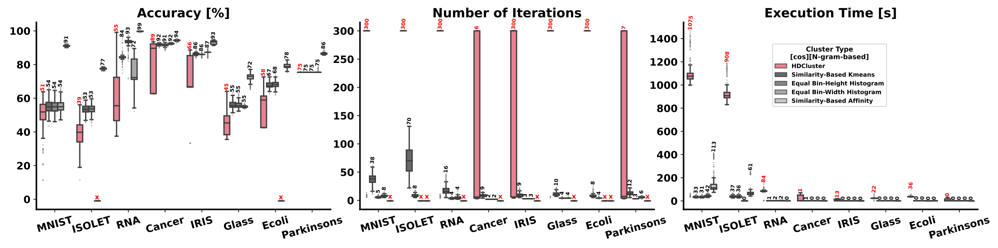
(a) Performance for clustering algorithms by binary HDC using record-based encoding.



(b) Performance for clustering algorithms by binary HDC using N-gram-based encoding.



(c) Performance for clustering algorithms by non-binary HDC using record-based encoding.



(d) Performance for clustering algorithms by non-binary HDC using N-gram-based encoding.

Figure 6.5: Comparison of proposed algorithms with HDCluster. For boxplots, the median values over 500 runs/trials are annotated on the top. The red values indicate the results for the baseline HDCluster. Our algorithms' results are in black. The symbol × implies the results are not available.

According to Figure 6.5, HDCluster reaches the maximum pre-specified iteration value (=300) more often than our proposed four algorithms. This observation can be more evident for non-binary HDC. Additionally, HDCluster nearly always requires a longer execution time—especially for MNIST and ISOLET datasets—in comparison to our proposed algorithms. This implies that the existing HDCluster may not perform a fast clustering for a dataset with large data samples (e.g., MNIST: 10,000, ISOLET: 7,797).

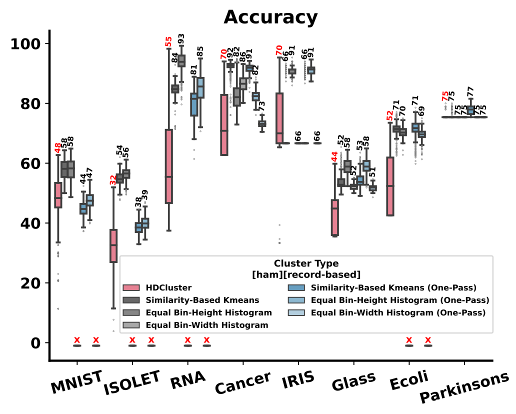
For our equal bin-width histogram clustering algorithm, the final cluster hypervectors are not evenly distributed in the measured range for eight datasets. In this approach, the number of samples for some bins is zero; this condition terminates the algorithm. Out of the eight datasets, four datasets are terminated for binary HDC; these include MNIST, ISOLET, RNA, and Ecoli. However, only ISOLET and Ecoli are terminated for non-binary HDC. This indicates that binary HDC is applicable in fewer cases than non-binary HDC.

From Figure 6.5, we observe that, for the RNA dataset, the binary HDC using N-gram-based encoding leads to the most non-robust accuracy with respect to the variance of the accuracy. A standard deviation for the clustering accuracy over 500 runs is used to further quantify the variance of the accuracy and the corresponding results are shown in Table 6.3. Binary HDC using N-gram-based encoding has ≥ 16.18 standard deviation of the accuracy performance among the baseline HDCluster and our proposed four HDC-based clustering algorithms for the RNA dataset. This reveals that the N-gram-based encoding for binary HDC can be significantly affected if the number of features n (=20,531) exceeds the dimensionality d (=10,000). Note that the variance of the accuracy using the non-binary HDC for this dataset is less.

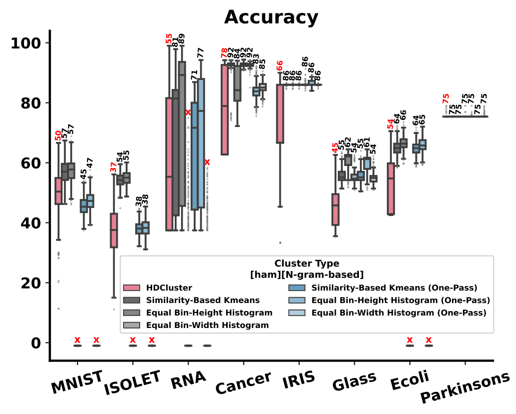
One-Pass Clustering

One-pass clustering refers to no iterative update of cluster hypervectors. To be more specific, here one-pass clustering only involves ❶-❷ of Figure 6.2. Without any doubt, one-pass clustering requires less execution time and fewer iterations for convergence as compared to the iterative update of the cluster hypervectors. For comparison purposes, only accuracy is considered.

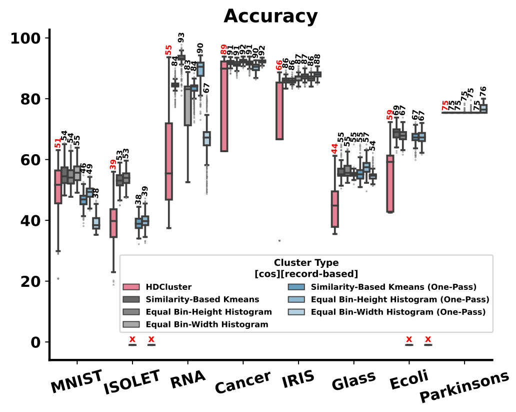
The experimental results for one-pass clustering are shown in Figure 6.6. Regarding the clustering accuracy, one-pass clustering algorithms are lower than the iterative update version of the proposed algorithms for four datasets (MNIST, ISOLET, RNA, and



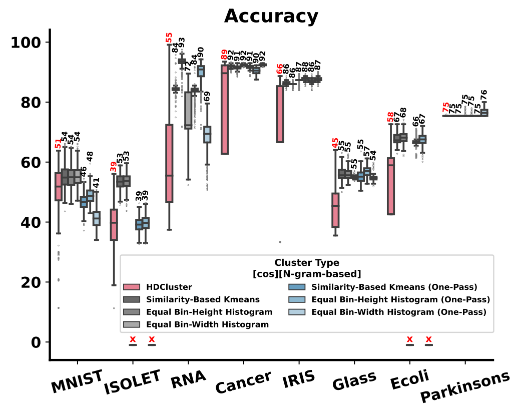
(a) Binary HDC using record-based encoding.



(b) Binary HDC using N-gram-based encoding.



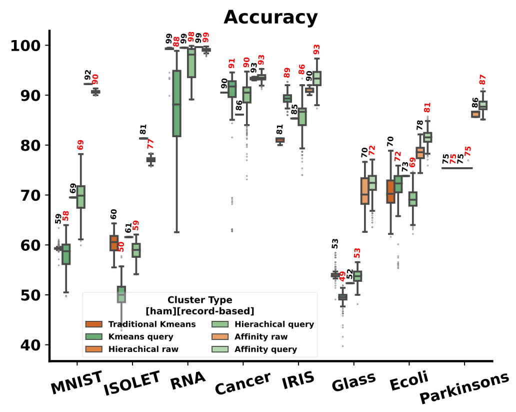
(c) Non-binary HDC using record-based encoding.



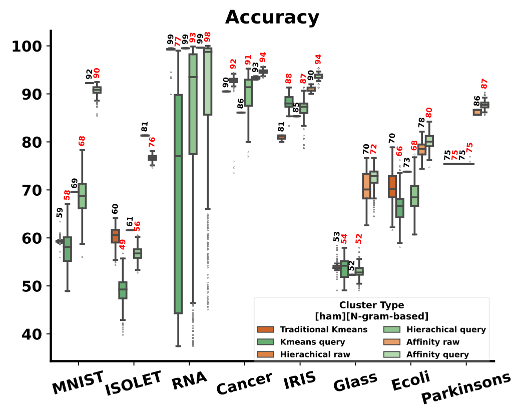
(d) Non-binary HDC using N-gram-based encoding.

Figure 6.6: Comparison of one-pass clustering with both the updated clusters and HDCluster. For boxplots, the median values over 500 runs/trials are annotated on the top. The red values indicate the results for the baseline HDCluster. Our algorithms' results are in black. The symbol × implies the results are not available.

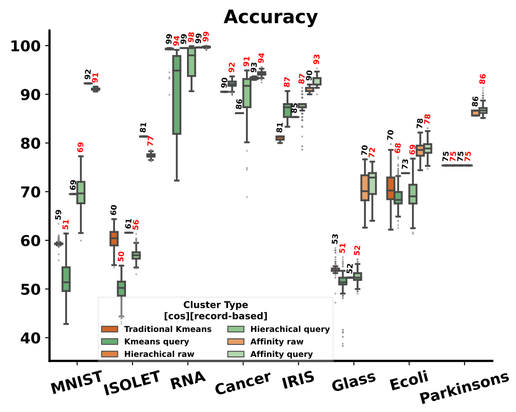
Cancer), while they can achieve comparable performance for the other four datasets. Additionally, compared to HDCluster, the variance of accuracy for one-pass clustering is more robust.



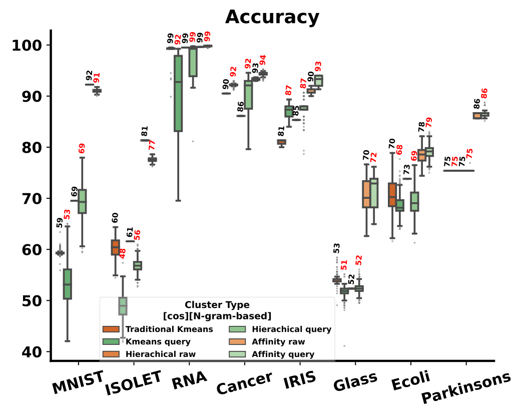
(a) Binary HDC using record-based encoding.



(b) Binary HDC using N-gram-based encoding.



(c) Non-binary HDC using record-based encoding.



(d) Non-binary HDC using N-gram-based encoding.

Figure 6.7: Comparison of clustering using original data and encoded data. For boxplots, the median values over 500 runs/trials are annotated on the top. Results for encoded data are in red, while those for the original data are in black.

6.4.3 Original Data Space vs Hyperdimensional Space

We now address the question of whether projection onto hyperdimensional space is helpful for data separation. We employ traditional k-means, hierarchical clustering, and affinity propagation toward both the original data and the encoded data (query hypervectors). Both binary and non-binary HDC, associated with record-based encoding and N-gram-based encoding, are employed. Therefore, all four possible cases are examined: binary/non-binary HDC with record-based/N-gram-based encoding. The corresponding experimental results are shown in Figure 6.7.

Based on Figure 6.7, the traditional clustering approach is better for at most three out of eight datasets, compared to HDC. To be more specific, (a) **Traditional K-means:** no projection is preferred for three datasets: ISOLET, RNA, and Glass if binary/non-binary HDC with record-based encoding, or non-binary HDC with N-gram-based encoding is employed. For binary HDC with N-gram-based encoding, the performance is similar for both cases. For the IRIS dataset, projection onto hyperdimensional space in all four possible cases achieves 6% improvement in accuracy as compared to the original space. In terms of the Cancer dataset, projection onto hyperdimensional space outperforms the original data by 2% accuracy over three cases, whereas it is comparable between original space and hyperdimensional space for binary HDC with record-based encoding. For the remaining three datasets, projection onto hyperdimensional space can achieve comparable performance with the original space. (b) **Hierarchical Clustering:** Both ISOLET and Ecoli datasets do not benefit from projection onto hyperdimensional space, as this may cause $2\% \sim 5\%$ accuracy degradation. Projection can achieve comparable performance for the other six datasets. Among them, IRIS and Cancer datasets benefit from projection as the accuracy is improved by approximately $2\% \sim 5\%$. (c) **Affinity Propagation:** Six out of eight datasets can achieve comparable or better accuracy if they are projected onto the hyperdimensional space. MNIST and ISOLET datasets experience an accuracy drop (of approximately 2% and 5%, respectively) for hyperdimensional projection.

The aforementioned observations are summarized in Table 6.2. To summarize, the projection of five datasets onto hyperdimensional space can lead to comparable or better accuracy performance compared to traditional k-means, hierarchical clustering, and affinity propagation. For all of these three traditional clustering algorithms, ISOLET

Table 6.2: Performance for three traditional clustering algorithms[1].

Datasets	MNIST	ISOLET	IRIS	Glass	RNA-seq	Cancer	Ecoli	Parkinson's
# of Data Samples (N)	10,000	7,797	150	214	801	569	336	195
# of Features (n)	784	617	4	9	20,531	30	7	22
# of Clusters (k)	10	26	3	6	5	2	8	2
Projection for k-means	✔	✘	✅	✘	✘	✅	✔	✔
Projection for Hierarchical Clustering	✔	✘	✅	✔	✔	✅	✘	✔
Projection for Affinity Propagation	✘	✘	✅	✔	✔	✅	✔	✅

[1]In last three rows, accuracy performance compared to original data: comparable (✔), significantly better (✅), and lower (✘).

performance is better for the original space, while IRIS performance is better for the hyperdimensional space. This indicates that projection onto hyperdimensional space might not be helpful for data separability when we have a large number of clusters, e.g., $k = 26$. Additionally, the projection onto hyperdimensional space can lead to higher performance when the number of clusters k and the number of features n are both small.

6.4.4 Further Discussion

The High/Robust Accuracy and Fast Convergence of Our Proposed HDC-based Clustering Algorithms

As mentioned in Sec. 6.1.2, the initial cluster hypervectors of HDCluster are random seed hypervectors, that are either ($\{0,1\}^d$) or ($\{-1,1\}^d$). It indicates there exist in total $\binom{2^d}{k}$ ways of initializing the cluster hypervectors, which leads to the non-robust clustering accuracy performance. Low accuracy is easily obtained if the positions of the initially assigned cluster hypervectors in the hyperdimensional space are all far away from the encoded query hypervectors, so that query hypervectors have the same similarity results as the cluster hypervectors and cannot be correctly separated/clustered. High accuracy could be achieved when the initially assigned k clusters have different similarity results with the query hypervectors. Our proposed HDC-based clustering algorithms assign the initial cluster hypervectors from the data domain. In other words, our algorithms utilize the information leveraged by the query hypervectors. As a result, the initial k cluster hypervectors are determined by query hypervectors in our algorithms and the source of the accuracy variance over 500 runs only comes from the inevitable randomness of seed hypervectors. Additionally, this data-domain-based assignment speeds up the convergence for the iterative update of center hypervectors.

Similarity-based Hierarchical Clustering is Excluded in Our Proposed HDC-based Clustering Algorithms

Similar to the similarity-based affinity propagation, we also feed the $N \times N$ similarity result into the hierarchical clustering. The corresponding performance is significantly lower than that of both the raw data and encoded data for hierarchical clustering (Sec. 6.4.3). As a result, similarity-based hierarchical clustering is not considered in our algorithms.

Binary HDC Shows Limitations as Compared to Non-binary HDC

Compared to non-binary HDC, binary HDC with equal bin-width histogram clustering cannot be applied to two more datasets. In binary HDC, the minimum change of cluster hypervectors is convergent within 0.02, whereas in non-binary HDC, it is 0.01. This reveals that non-binary HDC is more accurate than binary HDC, which confirms the existing knowledge of HDC [19, 38].

Applications of HDC-based Clustering

Several recent works [18, 132] indicate a trend of applying HDC to biological datasets that can obtain surprisingly great performance. Based on our experimental results over eight datasets (Figures. 6.5 and 6.7), HDC-based clustering can achieve high accuracy ($> 90\%$) in RNA and Cancer datasets. Therefore, the combination of HDC with biological datasets could provide interesting results and new insights.

Statistics of the clustering accuracy for HDCluster and our proposed HDC-based algorithms

Table 6.3 summarizes the mean and standard deviation results of the clustering accuracy over 500 runs for both the baseline HDCluster and our algorithms. The standard deviation reflects the variance of the accuracy. For each dataset, the highest performance is in bold.

Table 6.3: Performance comparison of our proposed algorithms and the baseline HDCluster[1].

Method	MNIST	ISOLET	RNA	Cancer	IRIS	Glass	Ecoli	Parkinson's
			Binary HDC using Record-based Encoding					
HDCluster	47.10(±8.79)	32.26(±8.11)	59.01(±16.05)	73.50(±10.44)	72.39(±13.85)	43.17(±6.07)	53.49(±10.80)	75.45(±0.47)
SB Kmeans	57.87(±3.35)	54.80(±2.13)	84.67(±2.41)	92.17(±2.31)	72.52(±10.39)	53.97(±2.73)	71.30(±2.04)	75.60(±0.78)
Bin Height	57.96(±3.36)	56.33(±1.94)	93.70(±3.37)	82.52(±4.73)	90.96(±1.35)	58.81(±2.40)	70.50(±1.60)	75.38(±0.00)
Bin Width	/	/	/	86.52(±2.54)	66.67(±0.00)	52.34(±0.94)	/	75.38(±0.00)
SB Kmeans (1-Pass)	44.70(±2.34)	38.60(±2.17)	79.75(±4.86)	91.57(±1.52)	72.21(±10.18)	54.26(±2.42)	71.43(±2.74)	77.59(±1.55)
Bin Height (1-Pass)	47.58(±2.59)	39.94(±2.20)	85.24(±4.68)	82.32(±1.87)	91.06(±1.28)	58.93(±2.39)	69.54(±1.72)	75.38(±0.00)
Bin Width (1-Pass)	/	/	/	73.19(±1.05)	66.67(±0.00)	51.73(±0.79)	/	75.38(±0.00)
SB Affinity	90.67(±0.26)	77.07(±0.46)	99.03(±0.34)	93.55(±0.68)	93.37(±1.76)	72.34(±2.33)	**81.65(±1.45)**	**87.92(±1.24)**
			Binary HDC using N-gram-based Encoding					
HDCluster	49.78(±7.86)	37.03(±7.96)	60.10(±21.26)	80.38(±12.39)	68.30(±17.70)	44.92(±6.36)	53.56(±8.20)	75.43(±0.33)
SB Kmeans	56.88(±3.77)	54.16(±2.20)	68.77(±20.36)	92.38(±2.60)	85.99(±0.14)	55.43(±2.18)	64.96(±2.30)	75.38(±0.00)
Bin Height	57.36(±3.82)	54.94(±2.24)	75.19(±23.94)	84.82(±6.21)	85.99(±0.14)	60.53(±3.16)	66.52(±2.23)	75.38(±0.00)
Bin Width	/	/	7.70(±18.84)	92.71(±0.61)	86.00(±0.11)	55.12(±1.97)	/	75.41(±0.26)
SB Kmeans (1-Pass)	45.55(±2.92)	37.94(±2.32)	64.14(±17.57)	92.61(±0.69)	86.13(±0.52)	55.73(±2.35)	64.77(±2.16)	75.41(±0.19)
Bin Height (1-Pass)	47.29(±2.94)	38.31(±2.63)	69.32(±20.79)	83.78(±2.58)	86.58(±0.85)	60.00(±2.71)	65.91(±2.35)	75.38(±0.00)
Bin Width (1-Pass)	/	/	7.33(±17.84)	85.09(±1.78)	86.00(±0.09)	54.91(±1.81)	/	75.46(±0.37)
SB Affinity	90.68(±1.01)	76.65(±0.59)	89.22(±16.18)	**94.62(±0.41)**	**93.73(±0.68)**	**72.70(±2.01)**	80.15(±1.53)	87.56(±0.94)
			Non-Binary HDC using Record-based Encoding					
HDCluster	50.26(±8.28)	38.97(±6.41)	58.32(±15.28)	80.15(±13.89)	68.68(±16.27)	44.12(±6.41)	55.46(±9.69)	75.38(±0.00)
SB Kmeans	54.95(±3.45)	53.04(±2.61)	84.42(±1.56)	91.90(±0.82)	85.88(±0.99)	55.84(±2.17)	68.58(±1.67)	75.38(±0.00)
Bin Height	54.45(±3.21)	53.76(±2.39)	93.05(±2.12)	91.32(±1.01)	85.77(±0.96)	56.35(±2.38)	67.93(±1.75)	75.38(±0.00)
Bin Width	55.51(±2.76)	/	77.77(±7.12)	92.32(±0.58)	86.83(±1.42)	55.15(±0.78)	/	75.44(±0.31)
SB Kmeans (1-Pass)	46.58(±2.25)	38.98(±2.12)	82.98(±2.57)	91.70(±0.87)	87.31(±0.98)	55.43(±2.12)	67.34(±1.58)	75.58(±0.56)
Bin Height (1-Pass)	49.22(±2.01)	39.82(±2.22)	89.04(±3.89)	90.28(±1.32)	86.85(±1.03)	57.36(±2.09)	67.38(±1.78)	75.38(±0.00)
Bin Width (1-Pass)	39.17(±2.44)	/	66.35(±6.50)	92.22(±0.53)	87.95(±1.04)	54.72(±1.46)	/	76.70(±1.23)
SB Affinity	91.07(±0.18)	77.48(±0.39)	99.67(±0.16)	94.30(±0.48)	93.00(±0.86)	72.37(±2.29)	78.92(±1.28)	86.78(±1.02)
			Non-Binary HDC using N-gram-based Encoding					
HDCluster	50.16(±9.00)	38.92(±7.35)	59.86(±17.24)	79.99(±13.95)	67.74(±16.83)	44.40(±6.41)	54.66(±10.14)	75.39(±0.02)
SB Kmeans	55.18(±3.63)	53.48(±2.50)	84.30(±1.58)	92.00(±0.45)	86.11(±0.65)	55.79(±1.72)	67.83(±1.56)	75.38(±0.00)
Bin Height	54.99(±3.57)	53.57(±2.50)	93.14(±2.49)	91.48(±0.67)	85.86(±0.65)	55.89(±1.82)	68.15(±1.77)	75.38(±0.00)
Bin Width	55.46(±3.27)	/	74.86(±6.50)	92.39(±0.26)	86.99(±1.13)	55.00(±0.47)	/	75.46(±0.38)
SB Kmeans (1-Pass)	46.66(±2.50)	39.20(±2.22)	83.31(±2.57)	91.75(±0.53)	87.48(±0.79)	55.22(±1.79)	66.79(±1.43)	75.44(±0.28)
Bin Height (1-Pass)	48.82(±2.64)	39.67(±2.54)	89.66(±3.52)	90.42(±1.09)	86.83(±0.73)	56.91(±1.72)	67.54(±1.87)	75.38(±0.00)
Bin Width (1-Pass)	41.27(±2.92)	/	67.90(±5.46)	92.39(±0.23)	87.80(±0.77)	54.77(±0.98)	/	76.53(±1.10)
SB Affinity	**91.08(±0.41)**	**77.53(±0.45)**	**99.72(±0.13)**	94.33(±0.35)	93.11(±0.79)	72.30(±2.52)	79.15(±1.27)	86.46(±0.42)

[1]This table displays the clustering accuracy over 500 runs in the format of [mean (± standard deviation)]. Symbol "/" indicates the result is not available.

6.5 Conclusion

We propose four HDC-based clustering algorithms based on the categorized information that take advantage of the encoded data—intra-cluster hypervectors have a higher similarity than inter-cluster hypervectors. We measure our algorithms by employing two standard encoding algorithms (record-based and N-gram-based encoding) for both binary and non-binary HDC. As compared to the existing HDCluster, our proposed HDC-based algorithms achieve better and more robust accuracy, fewer iterative updates of cluster hypervectors, and less execution time when tested over eight datasets. Similarity-based affinity propagation outperformed the other three HDC-based clustering algorithms on eight datasets by $2\% \sim 38\%$ in clustering accuracy. Even for one-pass clustering, our proposed algorithms can provide more robust clustering accuracy than HDCluster. In terms of whether to use the original data or encoded data, we find that five out of eight datasets that are projected onto hyperdimensional space can achieve better or comparable clustering accuracy. In particular, ISOLET does not require projection, whereas IRIS benefits from hyperdimensional projection. This observation implies that projecting onto hyperdimensional space is attractive when both the number of clusters k and the number of features n are small. Maintaining the original data space is recommended when the number of clusters, k, is large. Future work will be directed towards three avenues. First, all the discussed clustering problems in this work are provided with ground truth. Additionally, the target number of clusters k is already known. The capability of HDC to infer the optimal number of clusters k from a given dataset without ground truth should be investigated. Second, the algorithms in this research are more suitable for software implementations. Due to HDC's energy efficiency, future work should be directed towards hardware implementations. Third, we find that HDC-based clustering algorithms perform well in RNA and Cancer datasets. Therefore, future efforts should address clustering different types of biological datasets in the HDC domain to obtain better performance and gain new insights.

Chapter 7

Clustering and classification for brain stimulation with HDC

The study that presented in this chapter is currently being written [25, 26]. This chapter proposes a novel algorithm that utilizes HDCluster to determine the number of clusters for clinical trajectories. These trajectories are generated from individuals receiving repetitive transcranial magnetic stimulation (rTMS) treatment for major depressive disorder (MDD). In addition to clustering, this chapter also introduces a category prediction method that utilizes 34 measured cognitive variables to predict clinical response. The ultimate goal of this research is to investigate the applicability of HDC in analyzing the efficacy of rTMS treatment for MDD. Specifically, this chapter focuses on two specific aims: clustering and classification.

7.1 Introduction

Patients who suffer from major depressive disorder (MDD) experience persistent sadness and loss of interest, which severely influences the quality of their daily life. Transcranial magnetic stimulation (TMS), a noninvasive form of brain stimulation, can interfere with cognitive functions and has been applied to treat MDD. Despite achieving great success in treating MDD over the past few decades [133–135], TMS treatment presents several challenges, including high costs, inconvenience for patients who must visit daily for a month, slow effects that require repetitive treatments, and variable clinical responses.

Although TMS treatment can yield a comparable response rate to medication for MDD, the corresponding mechanism and its effect on cognition are not well understood. Identifying biomarkers that can predict clinical responses to TMS could lead to more efficient and personalized TMS treatments.

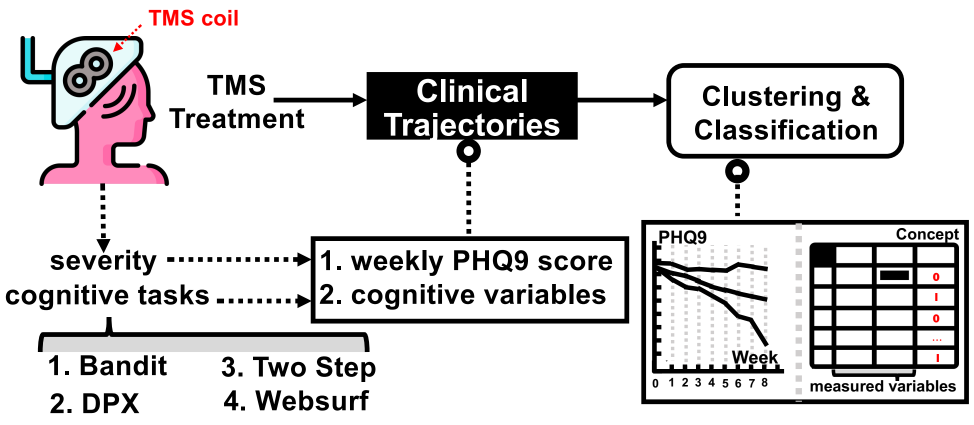


Figure 7.1: Goal overview.

To further understand the mechanism of TMS, as illustrated in Figure 7.1, patients who suffer from MDD need to receive repetitive TMS (rTMS) treatment with the aid of a TMS coil for several weeks. Four tasks that assess cognitive control—Bandit [136], Two Step [137–139], Dot-Probe Expectancy (DPX) [140–143] and Websurf [144–147]—should be conducted by patients. During the execution period, cognitive variables are measured in the clinic. Treatment information including the patient ID, week number, PHQ9 score[1], and cognitive variables are recorded for each treatment. Clinical trajectories can be obtained on a weekly basis. Two specific tasks are investigated by employing HDC in this chapter: (a) Explore the clinical response patterns exhibited by patients undergoing rTMS treatment. (**Task 1:** clustering). (b) Study the predictive capability of TMS response through measured cognitive variables (**Task 2:** classification).

For Task 1, we propose an algorithm based on HDC to estimate the number of clusters in a given system of clinical trajectories. This can be achieved by utilizing an existing HDC-based clustering algorithm—HDCluster [14]. Task 2 should ideally be a regression task that uses measured cognitive variables to predict the actual PHQ9 score. However, the regression models can only yield an average mean absolute error (MAE) of

[1]reflects the severity of MDD.

approximately 5. Instead of predicting the exact numerical values, the ability to predict the concept of clinical response of rTMS treatment is also meaningful clinically. This Task 2 is essentially a binary classification problem.

The contribution of this work can be summarized as follows: (a) The usage of the existing HDCluster [14] requires the number of target clusters to be already known. In this work, we develop an algorithm that statistically estimates/approximates the number of clusters of the given clinical trajectories. Our proposed algorithm repeats the HDCluster algorithm with different random seeds by giving all possible pre-defined numbers of clusters. To the best knowledge of the authors, our work is the first to determine the number of clusters using HDC-based clustering algorithms. (b) Due to the inherent noise in measuring both the PHQ9 score and cognitive variables, the achieved performance level of the regression model does not meet the standards of clinical satisfaction. To this end, we consider a category prediction task to predict the concept of clinical response for patients undergoing the rTMS treatment. The experimental results can achieve 0.81 AUC to predict on a weekly-observation basis, whereas it yields 0.86 AUC on a subject basis. (c) The motivation for investigating the clustering is enhanced. Clustering results of clinical trajectories vividly deliver the variability of TMS treatment across patients, especially the change rate of clinical responses. Our study indicates that the obtained clustering knowledge can be utilized for category prediction. In [], the concept of the clinical response is defined as logic “1” if the PHQ9 score is below 50% of the baseline; otherwise, it is defined as logic “0” to indicate no clinical response. In this chapter, we provide another criterion to define the concept of the clinical response using the clustering results. Such a criterion has also been demonstrated to be efficient in category prediction. (d) In order to improve the classification performance, we find the random seeds that are used to generate seed hypervectors could be considered as the fine-tuned parameters for HDC. This reveals that HDC is one of the approximate computing paradigms and its performance is not deterministic. (e) Our work demonstrates that HDC can be applied to cluster and classify the weekly track of the trajectories for TMS treatment.

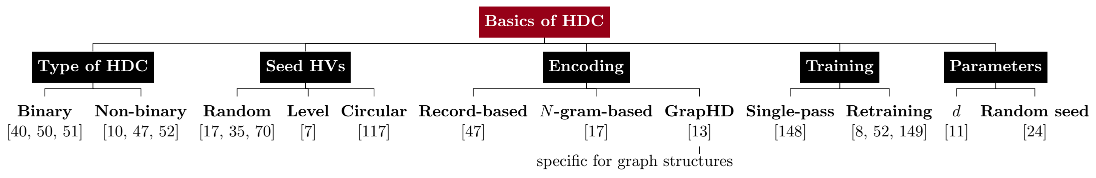


Figure 7.2: Overview of the basics of HDC.

7.2 Preliminaries

This section gives an overview of HDC and reviews two research problems using HDC—clustering and classification.

7.2.1 Basics of HDC

For any HDC system, the initial data samples are represented by ultra-long vectors—hypervectors. Such a process relies on *seed/atomic* hypervectors as the building blocks and then generates *compound/composite* hypervectors using an encoding algorithm. Seed hypervectors have three main categories: random (randomly generated so that they are orthogonal to each other), level (correction reserved), and circular (suitable for circular data: seasons of the year) hypervectors [117]. Typically, three operations—addition, multiplication, and permutation—can be involved in encoding algorithms. In contrast to classical computing, HDC relies on similarities between hypervectors rather than on actual values being computed. As sketched in Figure 7.2, a number of factors should be considered, but not limited to:

Types of HDC

The components of hypervectors can be binary (0, 1), bipolar (-1, 1), integers, real values, and complex values. HDC can be categorized as binary HDC and non-binary HDC. There are two main differences between them: (a) binary HDC requires the "*majority rule*" for bit-wise addition; (b) for the similarity measurement (δ) between hypervectors, as illustrated in Eq. (2.3), binary HDC uses the Hamming distance whereas non-binary HDC employs cosine similarity. Here **A**, **B** are two hypervectors and d represents their corresponding dimensionality.

In general, binary HDC is hardware-friendly since only unsigned logic "0" and "1".

For non-binary HDC, the accuracy performance is usually higher than binary HDC because there is no information loss by applying the majority rule.

In this chapter, non-binary HDC is restrictedly referred to as bipolar seed hypervectors and integer composite hypervectors, whereas binary HDC means both seed and composite hypervectors are binary hypervectors.

Encoding Algorithms

In HDC, two standard encoding algorithms are record-based encoding and N-gram-based encoding. They can encode images, signals, sequences, etc. One thing that should be mentioned is that GrapHD encoding [13] is used explicitly for graph structure data that contains nodes and edges.

Equation (7.1) summarizes the two standard encoding algorithms. As shown in Eq. (2a), both of them require the level hypervectors $\{\mathbf{L}_1, \cdots, \mathbf{L}_q\}$ to represent the quantized values $\bar{\mathbf{V}}_j$ for the raw data, where the quantization level is denoted by q. (a) Record-based encoding (Eq. (7.1b)) employs the random hypervectors $\{\mathbf{ID}_1, \cdots, \mathbf{ID}_q\}$ as the identifiers to encode the position information. (b) N-gram-based encoding (Eq. (7.1c)) considers the position information by permutation operations ($\rho(\cdot)$), where ρ^{n-1} means the hypervector is permuted $(n-1)$ times.

$$\bar{\mathbf{V}}_j \in \{\mathbf{L}_1, \cdots, \mathbf{L}_q\}, \text{ where } 1 \leq j \leq q. \tag{2a}$$

$$\mathbf{Query_H V}_i = \begin{cases} \bar{\mathbf{V}}_1 * \mathbf{ID}_1 + \cdots + \bar{\mathbf{V}}_n * \mathbf{ID}_n, & \text{(2b)} \\ \bar{\mathbf{V}}_1 + \rho\bar{\mathbf{V}}_2 + \cdots + \rho^{n-1}\bar{\mathbf{V}}_n. & \text{(2c)} \end{cases}$$

Training Mode

HDC supports two training modes for addressing classification tasks: single-pass and retraining. The HDC method, which only learns once for class hypervectors, is called single-pass learning. In retraining, the class hypervectors are iteratively updated from the training data until they meet the convergence condition [8, 52, 149]. As described in Eq. (7.3), if the **QueryHV** is misclassified as a wrong label W and should be correctly classified into label C, then the class hypervectors should be updated: (a) misclassified class hypervector(s) $\mathbf{ClassHV}_W$ subtract(s) the **QueryHV**; (b) target-correct class hypervector $\mathbf{ClassHV}_C$ adds the $\mathbf{QueryHV}$.

$$\begin{aligned}\mathbf{ClassHV}_C &= \mathbf{ClassHV}_C + \mathbf{QueryHV},\\ \mathbf{ClassHV}_W &= \mathbf{ClassHV}_W - \mathbf{QueryHV}.\end{aligned} \tag{7.3}$$

Fine-tuning Parameters in HDC

Dimensionality d and quantization level q are common fine-tuning parameters for HDC. The random seed to generate seed hypervectors has not been paid attention to in previous HDC studies for a long time but its significant impact on the robust clustering performance is emphasized in [24].

In our work, we consider all the factors/components mentioned above: two encoding algorithms, two types of hypervectors, and two training modes. Only $d = 10,000$ is studied.

HDCluster [14] is an HDC-based algorithm that mimics the traditional k-means algorithm [122] and has been shown to be more accurate than k-means across versatile machine-learning datasets. Note that the number of clusters for a given dataset should already be known before applying HDCluster. As described previously, Figure 6.1 illustrates an overview of HDCluster. A dataset, containing n data samples with f features, should be clustered into k groups. To represent the initial cluster centers, k random hypervectors are generated—$\mathbf{ClusterHV}_1, \cdots, \mathbf{ClusterHV}_k$. ❶ Each data sample is encoded using the record-based encoding method after the feature values have been quantized into q levels, where feature indices are encoded with $\mathbf{ID}$ hypervectors (random hypervectors), and feature values are encoded with $\mathbf{V} \in \{\mathbf{L}_1, \cdots, \mathbf{L}_q\}$ (level hypervectors)). ❷ In each iteration, every data sample is assigned to the cluster center that reflects the highest similarity and is tagged accordingly. ❸ Cluster hypervectors are updated by adding data samples associated with each cluster. ❹ Iterations will be terminated if (i) the center hypervectors change slightly between two consecutive iterations or (ii) the number of iterations exceeds the pre-defined number.

7.2.2 Classification using HDC

For classification, *class* hypervectors are trained during the training phase. To be more specific, the class hypervector is the summation of the hypervectors belonging to the same class. During the inference phase, the unknown data sample is encoded as a *query* hypervector. The corresponding label is determined by the highest similarity among the similarity results calculated by the query hypervector and the pre-trained class hypervectors. Readers are referred to [19] for more details.

7.3 Methodology

7.3.1 Clustering using HDC

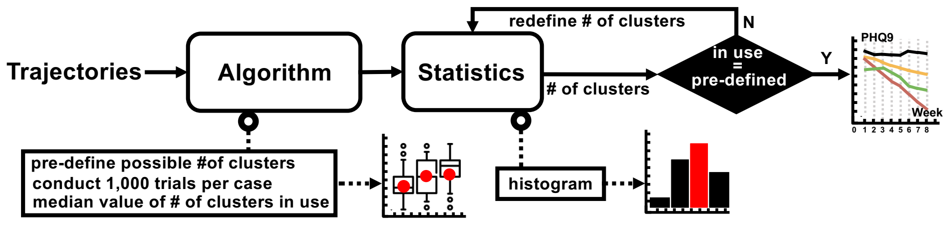


Figure 7.3: Clinical-trajectory-pattern clustering using HDC.

A schematic representation of the pipeline of HDC-based clustering of clinical trajectories is shown in Figure 7.3. All n trajectories are encoded as query hypervectors, which are denoted by $\{\mathbf{QueryHV}_1, \cdots, \mathbf{QueryHV}_n\}$. In Algorithm 1, statistical results are used to estimate the number of clusters for a given dataset, which contains n data samples. Therefore, theoretically, this dataset has a range of clusters between 1 and n. Pre-specify the number of clusters as i ($\in [1, n]$), and apply HDCluster for the given system. Repeat this procedure for $1,000$ runs and track the number of clusters practically in use. Note that the initial cluster hypervectors are generated with random seed j for the j^{th} run, where $1 \leq j \leq 1,000$. A boxplot is obtained, whose x-axis represents the number of pre-defined clusters and y-axis represents the number of clusters in use. The mode values of all boxes can generate a histogram, where the x-axis lists all possible numbers of clusters and the y-axis reflects the corresponding frequency. The highest bin statistically indicates the number of clusters for the given clinical trajectories.

Why statistical results of applying HDCluster can estimate the number of clusters for a given system? Based on different random seeds, the initial cluster hypervectors are generated at random, indicating that these starting points (cluster hypervectors)

Algorithm 1 The number of clusters analysis.

```
Require: A system of encoded hypervectors.
 1: for $i$ = 1:$n$, where $n$ is # of data samples do
 2:     Specify the number of clusters as $i$.
 3:     for $j$ = 1:1000 do
 4:         Generate $\mathbf{ClusterHV}_1, \cdots, \mathbf{ClusterHV}_i$.
 5:         Apply HDCluster to label all QueryHVs.
 6:         Calculate #of clusters in use.
 7:     end for
 8:     Record the mode value over 1000 runs.
 9: end for
10: Generate a histogram to display the frequency of all mode values. The highest bin
    reveals the number of clusters for this system.
11: if #of pre-defined clusters != #of clusters in use then
12:     Use the mode value of the system clusters as the #of pre-defined clusters.
13: end if
```

are equally likely to be selected from the hyperspace. It may be helpful to consider the main idea behind our Algorithm 1 as an analogy to the estimation of probability based on frequency statistics.

7.3.2 Category Prediction using HDC

Using 34 variables measured in four different cognitive tasks to predict the PHQ9 score, such a numerical regression problem practically leads to roughly 5 MAE. The noisiness of both PHQ9 scores and cognitive variables could cause such an unsatisfactory performance. Instead, we propose a category prediction strategy to predict the concept of clinical response after the TMS treatment. This category prediction is essentially a binary classification task, label "1" indicates there exists a clinical response after the TMS treatment while label "0" means no response.

Figure 7.4 illustrates the strategy of category prediction. A spreadsheet that contains the weekly records of different subjects is used as the input for this category prediction. In the preprocessing, ground truth (the correct "0" and "1" labels) should be predetermined for the raw data. Variables should be corrected/revised by subtracting its baseline[2] parameters per subject. Data are fed to an HDC-based classifier to conduct a

[2]Week 0 by default. Week 1 if Week 0 is missing.

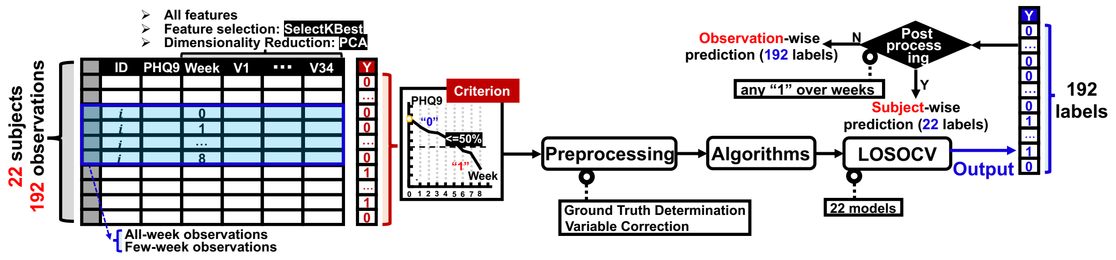


Figure 7.4: Category prediction overview.

leave-one-subject-out cross-validation (LOSOCV). If no post-processing is applied, it is an observation-wise prediction task; otherwise, a subject-wise prediction is conducted.

Ground Truth Determination

Two criteria are considered to determine the ground truth table.

Data in Use

Whether using the whole data in the spreadsheet is investigated in this work. For each subject, we use the all-week observations and few-week observations, separately. For the 35 features (week number and 34 cognitive variables), whether to use all features or selected features is also investigated. In terms of selected features, we use the SelectKBest algorithm for feature selection and principal component analysis (PCA) for dimensionality reduction.

Post-processing

The post-processing strategy in our research is, for a certain subject, if any clinical response "1" is detected over all eight weeks, then rTMS treatment is considered to be effective for this subject.

For category prediction, three cases are mainly studied: (a) all week-observations with criterion 1 (Case 1); (b) few week-observations with criterion 1 (Case 2); (c) few week observations with criterion 2 (Case 3).

7.4 Experimental Results

7.4.1 Materials

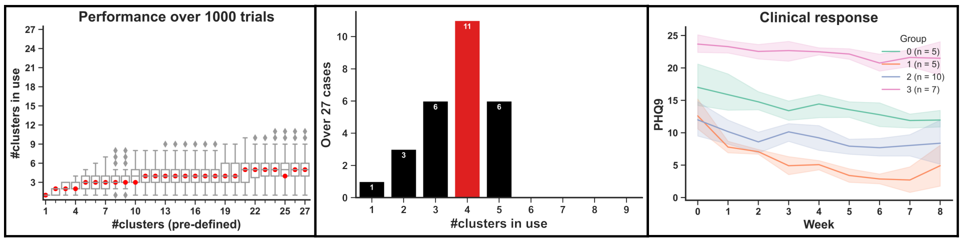
Thirty-two patients undergoing eight weeks of rTMS treatment for MDD were asked to complete four different cognitive tasks that assessed their cognitive control. During this process, a total of 34 cognitive variables are measured. To store the treatment information, patient ID, week number, PHQ9 score, and 34 cognitive variables are recorded in a spreadsheet. In week 0, no TMS treatment is applied; therefore week 0 is considered a baseline for further analysis. In this dataset, there are $n = 27$ patients who have consecutive eight-week PHQ9 scores for Task 1, whereas only $n = 22$ patients have the full 34 cognitive variables for Task 2.

For clustering, a large dataset ($n = 176$) is used to test our proposed algorithm to determine the number of clusters for a given system using HDCluster. In this dataset, patients receive 6 or 8 weeks of TMS treatment. Only PHQ9 scores are recorded. Since the 34 cognitive variables are not measured, this large dataset cannot be used for category prediction purposes.

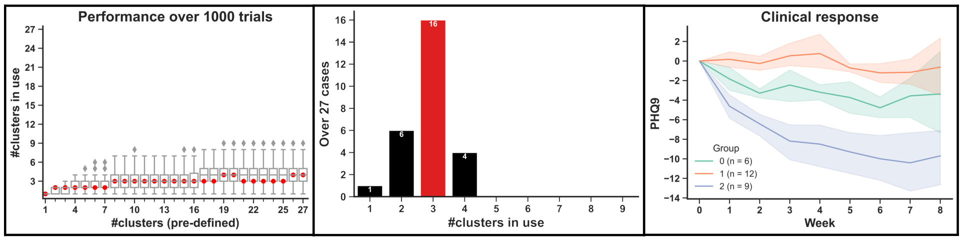
7.4.2 Clustering

Small Dataset

Figure 7.5 shows the clustering results using our proposed Algorithm 1 when $n = 27$ using N-gram-based encoding for non-binary HDC. For original clinical trajectories (Figure 7.5(a)), since the number of clinical trajectories (data samples) is 27 subjects, the number of clusters theoretically ranges from $[1, 27]$. We pre-define the number of clusters to apply Algorithm 1. The boxplot describes the number of used clusters over $1,000$ runs for all possible 27 cases of predefined clusters. The most frequent number of clusters in use (mode value per box) are marked by red dots. Among the 27 cases, the number of clusters in use is prone to be 4 as statistically revealed by the histogram. The corresponding clustering result using our proposed algorithm is shown in the third column. The last column shows a clustering result using the traditional clustering method, where the number of clusters is determined also by 4.



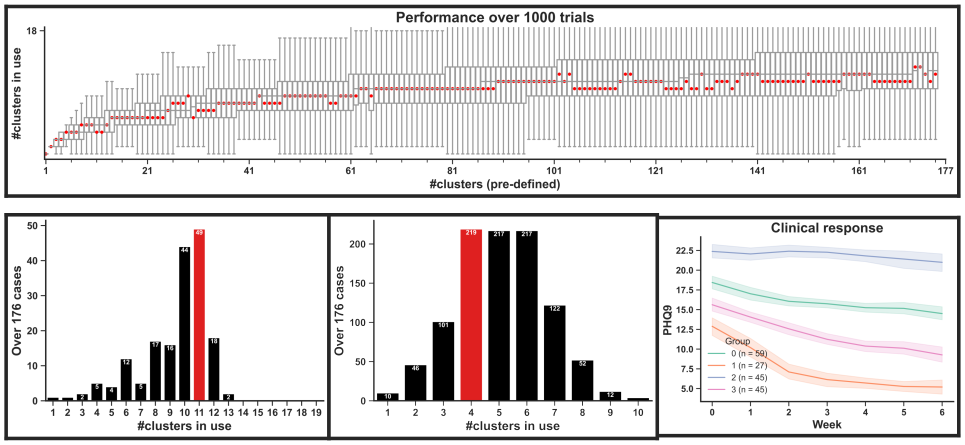
(a) Original trajectories clustering results using N-gram-based encoding for non-binary HDC.



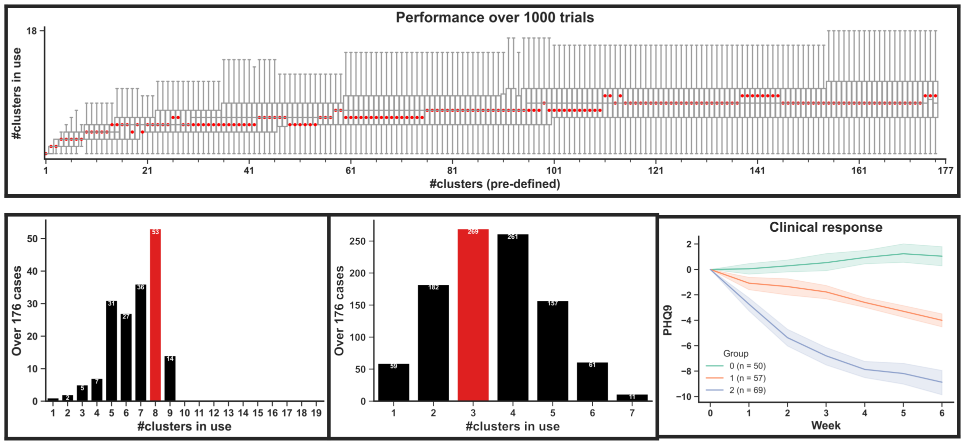
(b) Baseline-corrected results using N-gram-based encoding for non-binary HDC.

Figure 7.5: Clinical-trajectory-pattern clustering using HDC for a small dataset ($n = 27$).

Using original clinical trajectories, the clustering result essentially suggests the clusters of different levels of the PHQ9 scores: low, median, high, and top high. In clinics, the rate of clinical trajectories is more of interest. Similar to [], a baseline correction is conducted for the original data to subtract the baseline PHQ9 score per subject. Our HDC-based algorithm indicates there should be 3 clusters, which lines up with the traditional method. Based on Figure 7.5(b), the clustering results reveal that trajectories of patients who receive rTMS treatment are clustered as (i) no clinical change (ii) median rate of change and (iii) fast rate of change.



(a) Original trajectories clustering results using N-gram-based encoding for non-binary HDC.



(b) Baseline-corrected results using N-gram-based encoding for non-binary HDC.

Figure 7.6: Clinical-trajectory-pattern clustering using HDC for a large dataset ($n = 176$).

Large Dataset

Figure 7.6 displays the clustering patterns using a large dataset ($n = 176$) for both original and baseline-corrected trajectories. For original trajectories clustering, the boxplot in Figure 7.6(a) reveals that the number of clusters in use tends to be no more than 18. Using Algorithm 1 with N-gram-based encoding for non-binary HDC, the number of clusters for the given 176 trajectories is indicated to be 4. The corresponding clustering patterns reflect four clusters/levels of disorder severity. For baseline-corrected trajectories, 3 clustering patterns are indicated for the clinical response rates.

Table 7.1: LCMM results for original data ($n = 27$).

Model[1]	**G**[2]	**AIC**	**Membership [%]**							**MSE**
			group1	**group2**	**group3**	**group4**	**group5**	**group6**	**group7**	
Lin	1	1238.27	100.0							3.75
Lin	2	1241.26	66.67	33.33						3.81
Lin	3	1237.23	22.22	51.85	25.93					3.92
Lin	4	1224.73	48.15	14.81	11.11	25.93				3.95
Lin	5	1227.96	11.11	37.04	14.81	11.11	25.93			3.95
Lin	6	1228.23	11.11	33.33	14.81	11.11	3.70	25.93		4.07
Lin	7	1231.96	11.11	33.33	7.41	3.70	14.81	3.70	25.93	4.02
Quad	1	1198.04	100.0							2.14
Quad	2	1201.12	81.48	18.52						2.20
Quad	3	1208.68	22.22	37.04	40.74					2.25
Quad	4	1194.47	11.11	33.33	33.33	22.22				2.42
Quad	5	1189.21	37.04	7.41	25.93	3.70	25.93			2.61
Quad	6	1183.11	33.33	11.11	11.11	14.81	3.70	25.93		2.39
Quad	7	1173.39	18.52	18.52	7.41	22.22	7.41	14.81	11.11	2.35
Cub	1	1195.96	100.0							1.74
Cub	2	1197.18	85.19	14.81						1.63
Cub	**3**	**1184.96**	**44.44**	**33.33**	**22.22**					**1.93**
Cub	4	1222.99	0.00	25.93	74.07	0.00				1.75
Cub	5	1233.66	0.00	14.81	25.93	59.26	0.00			1.78
Cub	6	1220.96	0.00	0.00	44.44	33.33	22.22	0.00		1.93
Cub	7	1252.44	0.00	0.00	18.52	48.15	33.33	0.00	0.00	1.77

[1]Lin: linear; Quad: quadratic; Cub: cubic.
[2]G: number of clusters.

A classical algorithm, namely latent class mixed modeling (LCMM), is also applied to this clustering problem as a comparison with HDC. Here, heterogeneous linear mixed models are used, including "linear", "quadratic", and "cubic" formulas. Specifically, the 'hlme' function in the R language is employed. Table 7.1 reports the performance using LCMM for the original data when $n = 27$. The pre-specified number of clusters ranges from 1 to 7 because the maximum number of clusters identified by HDC is 7 as illustrated in Appendix A.1. The metrics to determine the number of clusters using LCMM include AIC[3], membership[4], and MSE for prediction. It is recommended that a good model have a low AIC, a low MSE, and a percentage of each group greater than 10%. On the basis of the lowest AIC, a quadratic model should be selected, which suggests seven

[3]short for Akaike's information criterion, which measures the quality of the model. The lower the AIC, the better performance the model achieves.

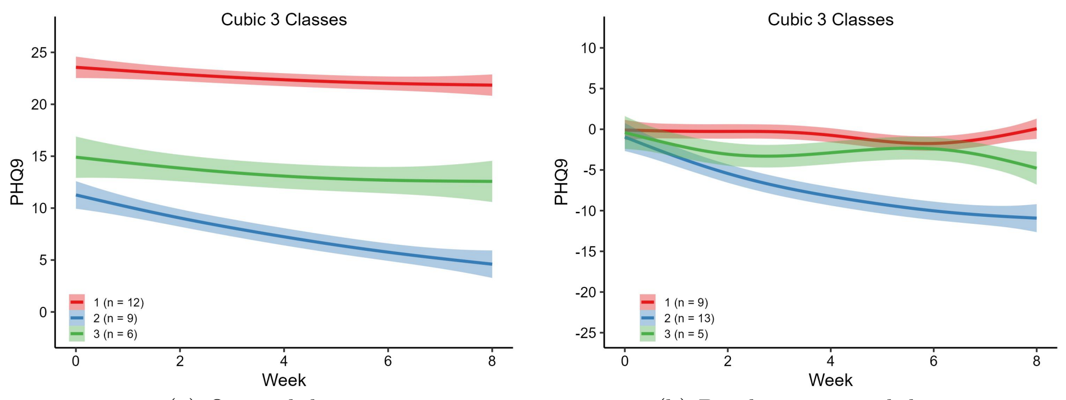
[4]reflects the percentage of each cluster in all data samples. In practical, "%group < 10%" should be avoided.

clusters. However, its membership exists at least one group is less than 10%. For the lowest MSE, a cubic model suggesting 2 clusters is the appropriate option; however, its AIC is relatively high. Considering all three metrics, a cubic model with 3 clusters is recommended by LCMM for original data.

Table 7.2: LCMM results for corrected data ($n = 27$).

Model	**G**	**AIC**	**Membership [%]**							**MSE**
			group1	**group2**	**group3**	**group4**	**group5**	**group6**	**group7**	
Lin	1	1178.20	100.0							3.99
Lin	2	1179.54	37.04	62.96						4.04
Lin	3	1184.81	25.93	33.33	40.74					4.04
Lin	4	1188.28	44.44	14.81	33.33	7.41				4.08
Lin	5	1192.18	18.52	3.70	29.63	40.74	7.41			4.21
Lin	6	1199.82	3.70	14.81	14.81	11.11	29.63	25.93		3.95
Lin	7	1202.84	3.70	14.81	14.81	11.11	11.11	37.04	7.41	4.06
Quad	1	1106.30	100.0							2.19
Quad	2	1106.48	25.93	74.07						2.26
Quad	3	1114.60	25.93	59.26	14.81					2.36
Quad	4	1117.78	25.93	3.70	59.26	11.11				2.35
Quad	5	1124.89	3.70	25.93	55.56	11.11	3.70			2.33
Quad	6	1130.33	14.81	11.11	25.93	29.63	7.41	11.11		2.37
Quad	7	1135.22	11.11	14.81	7.41	25.93	29.63	7.41	3.70	2.33
Cub	1	1097.89	100.0							1.68
Cub	2	1103.02	48.15	51.85						1.74
Cub	**3**	**1105.90**	**33.33**	**48.15**	**18.52**					**1.64**
Cub	4	1111.58	0.00	18.52	77.78	3.70				1.75
Cub	5	1123.56	0.00	18.52	3.70	77.78	0.00			1.75
Cub	6	1135.56	0.00	0.00	18.52	77.78	3.70	0.00		1.75
Cub	7	1147.56	0.00	0.00	18.52	3.70	77.78	0.00	0.00	1.75

Table 7.2 summarizes the performance of LCMM for baseline-corrected data when $n = 27$. Similarly, the optimal choice is the cubic model which suggests 3 clusters. Figure 7.7 shows the clustering patterns using LCMM. Observe that the patterns reflect the severity of MDD for original data, and the rates of change for baseline-corrected data. As with the HDC results, the clustering trajectories follow a similar pattern, although the final number of clusters may vary. When $n = 176$, LCMM suggests a cubic model with 3 clusters for original data, while it suggests a cubic model with 2 clusters for baseline-corrected data. Please refer to Appendix A.2 for more details.



(a) Original data. (b) Baseline-corrected data.

Figure 7.7: Trajectory clustering for LCMM when $n = 27$.

Table 7.3: Experimental results for category prediction using HDC.

Single-Pass	Observation					Subject				
Method[1]	ACC	SEN	SPEC	AUC	Parameters[2]	ACC	SEN	SPEC	AUC	Parameters
All Features										
Case 1	70.31	45.71	75.80	0.61	[Enc1, B, '4' '21']	59.09	52.63	100.00	0.76	[Enc1, B, '6' '21']
Case 2	55.88	83.33	47.44	0.65	[Enc2, B, '2' '21']	54.55	47.37	100.00	0.74	[Enc2, B, '0' '21']
Case 3	62.75	70.45	56.90	0.64	[Enc2, B, '2' '21']	59.09	50.00	100.00	0.75	[Enc1, B, '4' '21']
SelectedKBest										
Case 1	78.12	80.00	77.71	0.79	[Enc2, B, '1' '1' '2' '21']	72.73	62.50	100.00	0.81	[Enc2, B, '4' '1' '3' '21']
Case 2	77.45	83.33	75.64	0.79	[Enc2, B, '6' '1' '4' '21']	68.18	56.25	100.00	0.78	[Enc1, B, '3' '1' '5' '21']
Case 3	75.49	59.09	87.93	0.74	[Enc2, B, '5' '1' '1' '21']	72.73	60.00	100.00	0.80	[Enc2, NB, '5' '1' '10' '21']
PCA										
Case 1	60.42	85.71	54.78	0.70	[Enc1, NB, '1' '1' '21']	63.64	55.56	100.00	0.78	[Enc2, B, '9' '2' '21']
Case 2	57.84	79.17	51.28	0.65	[Enc2, B, '7' '21' '21']	54.55	47.37	100.00	0.74	[Enc1, B, '3' '3' '21']
Case 3	67.65	65.91	68.97	0.67	[Enc2, B, '7' '19' '21']	72.73	60.00	100.00	0.80	[Enc1, NB, '0' '8' '21']
Retraining	Observation					Subject				
Method	ACC	SEN	SPEC	AUC	Parameters	ACC	SEN	SPEC	AUC	Parameters
All Features										
Case 1	80.73	14.29	95.54	0.55	[Enc1, B, '1' '21']	54.55	50.00	58.33	0.54	[Enc1, NB, '7' '21']
Case 2	68.63	29.17	80.77	0.55	[Enc2, NB, '1' '21']	63.64	100.00	61.90	0.81	[Enc2, B, '0' '21']
Case 3	65.69	56.82	72.41	0.65	[Enc2, B, '9' '21']	63.64	52.94	100.00	0.76	[Enc2, B, '9' '21']
SelectedKBest										
Case 1	**83.33**	**77.14**	**84.71**	**0.81**	**[Enc1, B, '6' '1' '2' '21']**	72.73	100.00	66.67	0.83	[Enc2, B, '0' '1' '7' '21']
Case 2	77.45	58.33	83.33	0.71	[Enc2, NB, '5' '1' '2' '21']	**77.27**	**100.00**	**72.22**	**0.86**	**[Enc1, B, '9' '1' '30' '21']**
Case 3	73.53	68.18	77.59	0.73	[Enc1, B, '4' '1' '15' '21']	81.82	69.23	100.00	0.85	[Enc1, NB, '1' '1' '9' '21']
PCA										
Case 1	66.15	68.57	65.61	0.67	[Enc1, B, '0' '22' '21']	59.09	52.63	100.00	0.76	[Enc1, B, '0' '7' '21']
Case 2	76.47	50.00	84.62	0.67	[Enc1, NB, '2' '5' '21']	63.64	100.00	61.90	0.81	[Enc2, B, '8' '15' '21']
Case 3	68.63	63.64	72.41	0.68	[Enc2, B, '9' '18' '21']	77.27	64.29	100.00	0.82	[Enc2, NB, '1' '12' '21']

[1]Case 1: all week-observations with criterion 1; Case 2: few week-observations with criterion 1; Case 3: few week-observations with criterion 2.

[2]Enc1: record-based; Enc2: permutation; B: binary HDC; NB: non-binary HDC. For selected criteria of SelectKBest, '0': f_classif, '1': mutual_info_classif.

7.4.3 Category Prediction

As mentioned in Sec. 7.3.2, we consider the observation-based and subject-based category prediction. The first case predicts the efficacy of TMS treatment for a patient on a weekly basis, while the second case considers all predicted weekly results to make a final decision on the effectiveness of TMS treatment. Specifically, if any “1” is detected among all predicted week labels, the TMS treatment is considered effective in producing a clinical response.

Apart from using all features, a small number of features is considered by feature selection or dimensionality reduction. For feature selection, we use the SelectKBest algorithm with two selection criteria: “f_classif” (measures the F values) and “mutual_info_classif” (measures the mutual information) in the *scikit learn* library [150]. For dimensionality reduction, we consider PCA. Since we have in total of 35 features, we sweep the small number of features from 1 to 34.

Regarding HDC, we normalized the raw data into [0, 1], the quantization level is specified as $q = 21$, and the random seed ranges from [0, 9].

Table 7.3 lists the simulation results for category prediction using HDC. Note that for category prediction, we use 35 features (34 measured cognitive variables and week number) to predict the concept of clinical response. The information in the “parameters” column is listed in the order of (a) encoding algorithms, types of HDC, random seed, and quantization level for all features; (b) encoding algorithms, types of HDC, random seed, selection criteria, k selected features and quantization level for SelectKBest, (c) encoding algorithms, types of HDC, random seed, k components and quantization level for PCA.

According to this Table 7.3, (a) the best performance for subject-based category prediction is 0.86 AUC, which is higher than that for observation-based category prediction by 0.05. (b) Retraining can lead to optimal performance in both observation-based and subject-based category prediction scenarios. (c) In seven out of nine cases, predictions based on subjects perform better than predictions based on week-wise observations. This can be attributed to the fact that predicting categories based on observations requires a higher level of accuracy for weekly predictions as compared to the subject-wise category prediction which makes one prediction using all eight weeks’ results. (d) Feature engineering is proven to be efficient in improving category prediction since not all features contain useful information. For this particular research task, SelectKBest is sufficient and its optimal performance is achieved when the feature selection criterion based on mutual information is adopted. (e) This table also reveals that non-binary HDC cannot always guarantee a higher performance than binary HDC. Additionally, the optimal performance is achieved by different random seeds. The observation implies that HDC performance may not be deterministic or robust and could be approximate or stochastic. However, it is widely believed in the literature that HDC is resistant to noise since the information is uniformly distributed across all d bits [130].

Table 7.4: Comparison with SVM.

	Observation					Subject				
Method	**ACC**	**SEN**	**SPEC**	**AUC**	**Parameters**	**ACC**	**SEN**	**SPEC**	**AUC**	**Parameters**
SelectKBest										
Case1	85.94	57.14	92.36	0.75	1000, 0.1, [3 0 6], RBF	86.36	100	80	0.9	2, 0, 'scale', [0 1 1], Sigmoid
	83.33	**77.14**	**84.71**	**0.81**	**[Enc1, B, '6' '1' '2' '21'], Retraining**	72.73	100.00	66.67	0.83	[Enc2, B, '0' '1' '7' '21'], Retraining
Case2	86.27	62.50	93.59	0.78	1000, 'scale', [0 0 10], RBF	**90.91**	**88.89**	**92.31**	**0.91**	**1, 0.01, 'scale', [0 0 2], Sigmoid**
	77.45	83.33	75.64	0.79	[Enc2, B, '6' '1' '4' '21'], Single-pass	77.27	100.00	72.22	0.86	[Enc1, B, '9' '1' '30' '21'], Retraining
Case3	81.37	65.91	93.10	0.80	0.5, -0.01, 'scale', [2 1 2], Sigmoid	90.91	88.89	92.31	0.91	0.5, -0.01, 'scale', [2 1 2], Sigmoid
	75.49	59.09	87.93	0.74	[Enc2, B, '5' '1' '1' '21'], Single-pass	77.27	64.29	100.00	0.82	[Enc2, NB, '1' '12' '21'], Retraining

We applied SVM with the kernel of linear, RBF, and sigmoid functions to the same dataset for three cases. Table 7.4 compares the performance of HDC with the traditional SVM. Only the optimal results for each case are listed. We find: (a) HDC achieves comparable performance with SVM in observation-wise category prediction, whereas traditional SVM outperforms HDC for subject-wise category prediction by 0.05 AUC. (b) Subject-wise category prediction can achieve the highest performance (0.91 AUC), a 10% improvement over observation-wise category prediction. (c) For both SVM and HDC, the optimal performance is achieved when SelectKBest is applied across three cases.

7.4.4 Further Discussion

Importance/Motivation of Clustering

The experimental results for Case 3 in Table 7.3 give us a piece of strong evidence that the knowledge of the clustering patterns for the baseline-corrected data can be harnessed for category prediction (criterion 2 in Figure 7.4). Additionally, for a few weeks of observation, the clustering results show a downward trend for PHQ9 scores from weeks 4 to 7, followed by a rise at week 8. As a result, the week 8 observation is excluded from the analysis.

Length of Trajectories

For time-series clustering, classic algorithms even support unequal-length trajectory clustering [151]. In this work, only equal-length trajectories are studied by HDC.

Cost of Our Proposed Algorithm 1

Given n data samples, over $j = 1,000$ experiments are conducted even for a single predefined number of clusters. The complexity of this algorithm is measured by $\mathcal{O}(jn)$. Ideally, the greater the number of experiments (j) we conduct, the more accurate the approximation for the number of clusters. To determine the number of clusters, there should be a more elegant method that has less complexity/cost.

7.5 Conclusion

This chapter addresses two research tasks using HDC: (i) clustering for the clinical trajectories after TMS treatment and (ii) category prediction using 34 measured cognitive variables.

We have developed an algorithm for clustering that utilizes the existing HDCluster to derive statistical data for estimating or approximating the number of clusters in a given system. We test our proposed algorithm for a small $n = 27$ and a large $n = 176$. The group of clinical trajectories is suggested to have four clusters for original data and three clusters for baseline-corrected data. The corresponding clustering patterns follow a similar pattern as those indicated by the classic LCMM, however, the number of clusters varies.

Our analysis includes both observation-wise and subject-wise category prediction. The former scenario yields an AUC of 0.81, while the latter one achieves an AUC of 0.86. Improving performance in binary classification has been shown to be efficient through (a) feature engineering, particularly feature selection using SelectKBest; (b) retraining technique for HDC; (c) random seeds for the generation of seed hypervectors in HDC; (d) tailoring the classification goal while still ensuring its significance. This means, for subject-wise category prediction, we do not require as high an accuracy as that required for observation-wise category prediction. The efficiency of recovery following rTMS treatment is still measurable.

Future work will be directed towards three avenues. First, a more elegant HDC-based algorithm to determine the number of clusters with less computational complexity should be developed. Additionally, without loss of generality, it is important to test the statistical estimation of the clusters of a given system across various machine-learning datasets. Second, this work only investigates equal-length trajectory clustering. Further research could be focused on another practical scenario of clustering trajectories with unequal lengths. Third, it is important to identify applications in which the performance is not significantly affected by the random seeds used for seed hypervector generators.

Chapter 8

Conclusion and Future Directions

This dissertation investigates the applicability of HDC to biosignal-centric tasks, such as seizure detection and prediction (iEEG data), brain graph classification (fMRI data), and TMS treatment analysis (brain stimulation measurement). In this chapter, we summarize the key findings of our study and identify potential avenues for future research.

8.1 Key Findings

Below is a summary of the key findings.

- Our research contributes to the growing body of evidence supporting the potential of HDC for seizure detection, as demonstrated in previous studies [9, 16, 95]. In contrast to these studies, we use the Kaggle dataset [28] to evaluate the performance of HDC in this application. Our experimental results indicate that PSD encoding outperforms LBP encoding for seizure detection using HDC. Interestingly, we find that even a low dimensionality of $d = 100$ achieves good performance, highlighting the efficiency of HDC for this task.

- We uncover essential insights regarding the efficiency of PSD and LBP encoding methods for seizure detection and prediction. Specifically, we find that PSD and LBP encoding demonstrate high efficacy for seizure detection tasks. However, when it comes to seizure prediction, these encoding methods fall short in terms of their predictive capabilities. These findings underscore the need for further research and development of more advanced encoding techniques tailored explicitly for seizure prediction applications.

- Our study highlights the potential of GrapHD encoding for brain graph classification, which outperforms the existing HDC encoding algorithms, including record-based and N-gram-based encoding. Notably, GrapHD encoding is more efficient in terms of memory requirements.

- We emphasize the impact of random seed on the clustering performance of HDC, revealing that the existing HDCluster algorithm may not be as robust as expected. To address this limitation, we propose more reliable clustering algorithms that leverage the unique characteristics of hypervectors. Specifically, we utilize similarity measurements to group similar hypervectors and distinguish less similar ones, resulting in more accurate and robust clustering performance. These findings have important implications for the development and application of HDC in various domains.

- We propose one method of manipulating HDCluster to determine the number of clusters for given clinical trajectories for rTMS. We have demonstrated that category prediction (classification) is more accurate than numerical value prediction (regression) for TMS treatment analysis.

8.2 Future Outlook

The future would be oriented towards:

- Instead of LBP and PSD features, more efficient features for seizure prediction should be explored to enhance the performance.

- We trained our seizure detection and prediction model on an individual basis, which means that it is subject-specific. However, we recognize the importance of exploring group-based analysis for seizure detection and prediction, which can potentially enhance the scalability and generalizability of our models. Furthermore, given the diverse range of epilepsy types and presentations, it is crucial to consider the unique characteristics of each subtype when developing predictive models. By gaining a deeper understanding of the complexities of epilepsy and utilizing a range of modeling approaches, we can advance our ability to detect and predict seizures accurately, ultimately improving the quality of life for individuals with epilepsy.

- While HDC is capable of encoding higher dimension information, feature engineering remains a crucial step in the development of effective HDC classifiers when fewer resources are available. Our research on seizure detection, for instance, has demonstrated that a carefully selected set of three features is sufficient for achieving high performance on the Kaggle dataset. This highlights the importance of identifying the most relevant features for a given task and developing an efficient pipeline for feature selection. Therefore, it is recommended to explore efficient features for HDC classifiers.

- We applied the GrapHD encoding to brain graph classification with reduced memory requirements as compared to the other two HDC-based encoding approaches. Although we have successfully completed the encoding phase for classification, there is potential for further exploration of the GrapHD representation in the decoding phase. By extracting information from the graph structure, we can gain insights into the most distinct brain regions for different brain states, which can enhance our understanding of brain function and facilitate the development of more targeted interventions for neurological disorders.

- More “cognition” aspects of HDC, including analogical reasoning, relationship representation, and analysis, should be further developed.

- One of the key strengths of HDC lies in its energy efficiency. Simple bit-wise manipulations and the ability to enable parallel operations make HDC an attractive option for developing hardware platforms and architecture. To fully realize the benefits of HDC in practical applications, it is important to investigate specific hardware implementations and acceleration techniques for a given task.

References

[1] Abbas Rahimi, Tony F Wu, Haitong Li, Jan M Rabaey, H-S Philip Wong, Max M Shulaker, and Subhasish Mitra. Hyperdimensional computing nanosystem. *arXiv preprint arXiv:1811.09557*, 2018.

[2] Pentti Kanerva. *Sparse Distributed Memory.* MIT press, 1988.

[3] Sohum Datta, Ryan AG Antonio, Aldrin RS Ison, and Jan M Rabaey. A programmable hyper-dimensional processor architecture for human-centric IoT. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(3):439–452, 2019.

[4] Dominic Widdows and Trevor Cohen. Reasoning with vectors: A continuous model for fast robust inference. *Logic Journal of the IGPL*, 23(2):141–173, 2015.

[5] Abbas Rahimi, Pentti Kanerva, and Jan M Rabaey. A robust and energy-efficient classifier using brain-inspired hyperdimensional computing. In *Proceedings of the 2016 International Symposium on Low Power Electronics and Design*, pages 64–69. ACM, 2016.

[6] Abbas Rahimi, Simone Benatti, Pentti Kanerva, Luca Benini, and Jan M Rabaey. Hyperdimensional biosignal processing: A case study for EMG-based hand gesture recognition. In *2016 IEEE International Conference on Rebooting Computing (ICRC)*, pages 1–8. IEEE, 2016.

[7] Mohsen Imani, Chenyu Huang, Deqian Kong, and Tajana Rosing. Hierarchical hyperdimensional computing for energy efficient classification. In *2018 55th*

ACM/ESDA/IEEE Design Automation Conference (DAC), pages 1–6. IEEE, 2018.

[8] Mohsen Imani, Deqian Kong, Abbas Rahimi, and Tajana Rosing. VoiceHD: Hyperdimensional computing for efficient speech recognition. In *2017 IEEE International Conference on Rebooting Computing (ICRC)*, pages 1–8. IEEE, 2017.

[9] Alessio Burrello, Lukas Cavigelli, Kaspar Schindler, Luca Benini, and Abbas Rahimi. Laelaps: An energy-efficient seizure detection algorithm from long-term human iEEG recordings without false alarms. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 752–757. IEEE, 2019.

[10] Justin Morris, Mohsen Imani, Samuel Bosch, Anthony Thomas, Helen Shu, and Tajana Rosing. Comphd: Efficient hyperdimensional computing using model compression. In *2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 1–6. IEEE, 2019.

[11] Mohsen Imani, Samuel Bosch, Mojan Javaheripi, Bita Rouhani, Xinyu Wu, Farinaz Koushanfar, and Tajana Rosing. Semihd: Semi-supervised learning using hyperdimensional computing. In *IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–8, 2019.

[12] Alec Xavier Manabat, Celine Rose Marcelo, Alfonso Louis Quinquito, and Anastacia Alvarez. Performance analysis of hyperdimensional computing for character recognition. In *2019 International Symposium on Multimedia and Communication Technology (ISMAC)*, pages 1–5. IEEE, 2019.

[13] Prathyush Poduval, Ali Zakeri, Farhad Imani, Haleh Alimohamadi, and Mohsen Imani. Graphd: Graph-based hyperdimensional memorization for brain-like cognitive learning. *Frontiers in Neuroscience*, page 5, 2022.

[14] Mohsen Imani, Yeseong Kim, Thomas Worley, Saransh Gupta, and Tajana Rosing. HDCluster: An accurate clustering using brain-inspired high-dimensional computing. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1591–1594. IEEE, 2019.

[15] Pentti Kanerva. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive computation*, 1(2):139–159, 2009.

[16] Alessio Burrello, Kaspar Schindler, Luca Benini, and Abbas 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)*, pages 1–4. IEEE, 2018.

[17] Aditya Joshi, Johan T Halseth, and Pentti Kanerva. Language geometry using random indexing. In *International Symposium on Quantum Interaction*, pages 265–274. Springer, 2016.

[18] Mohsen Imani, Tarek Nassar, Abbas Rahimi, and Tajana Rosing. HDNA: Energy-efficient DNA sequencing using hyperdimensional computing. In *2018 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*, pages 271–274. IEEE, 2018.

[19] Lulu Ge and Keshab K Parhi. Classification using Hyperdimensional computing: A review. *IEEE Circuits and Systems Magazine*, 20(2):30–47, 2020.

[20] Lulu Ge and Keshab K Parhi. Seizure detection using power spectral density via hyperdimensional computing. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7858–7862. IEEE, 2021.

[21] Lulu Ge and Keshab K Parhi. Applicability of hyperdimensional computing to seizure detection. *IEEE Open Journal of Circuits and Systems*, 3:59–71, 2022.

[22] Lulu Ge and Keshab K Parhi. Hyperdimensional computing cannot predict seizures using lbp and psd features from ieeg. 2023. ***[Accepted]***.

[23] Lulu Ge, Ali Payani, Hugo Latapie, and Keshab K Parhi. Classifying functional brain graphs using graph hypervector representation. *2023 Asilomar Conference on Signals, Systems, and Computers*, 2023. ***[Accepted]***.

[24] Lulu Ge and Keshab K Parhi. Robust clustering using hyperdimensional computing.
arXiv preprint arXiv:2312.02407, 2023.

[25] Lulu Ge, Aaron McInnes, Alik S Widge, and Keshab K Parhi. Determining the number of clusters of clinical response of transcranial magnetic stimulation treatment using hyperdimensional computing. 2024. ***[Will Be Submitted]***.

[26] Lulu Ge, Aaron McInnes, Alik S Widge, and Keshab K Parhi. Classification of clinical response of transcranial magnetic stimulation treatment using hyperdimensional computing. 2024. ***[Will Be Submitted]***.

[27] The SWEC-ETHZ iEEG database. `http://ieeg-swez.ethz.ch/`.

[28] Upenn and mayo clinic's seizure detection challenge. `https://www.kaggle.com/c/seizure-detection/data`.

[29] Paul Smolensky. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial intelligence*, 46(1-2):159–216, 1990.

[30] Tony A Plate. Holographic reduced representations. *IEEE Transactions on Neural networks*, 6(3):623–641, 1995.

[31] Pentti Kanerva et al. Fully distributed representation. *PAT*, 1(5):10000, 1997.

[32] Dmitri A Rachkovskij and Ernst M Kussul. Binding and normalization of binary sparse distributed representations by context-dependent thinning. *Neural Computation*, 13(2):411–452, 2001.

[33] Ross W Gayler. Multiplicative binding, representation operators & analogy (workshop poster). 1998.

[34] Kenny Schlegel, Peer Neubert, and Peter Protzel. A comparison of vector symbolic architectures. *arXiv preprint arXiv:2001.11797*, 2020.

[35] Michael Hersche, José del R Millán, Luca Benini, and Abbas Rahimi. Exploring embedding methods in binary hyperdimensional computing: A case study for

motor-imagery based brain-computer interfaces. *arXiv preprint arXiv:1812.05705*, 2018.

[36] Abbas Rahimi, Sohum Datta, Denis Kleyko, Edward Paxon Frady, Bruno Olshausen, Pentti Kanerva, and Jan M Rabaey. High-dimensional computing as a nanoscalable paradigm. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 64(9):2508–2521, 2017.

[37] Randal E Bryant, O'Hallaron David Richard, and O'Hallaron David Richard. *Computer systems: a programmer's perspective*, volume 2. Prentice Hall Upper Saddle River, 2003.

[38] Agnieszka Patyk-Łońska, Marek Czachor, and Diederik Aerts. A comparison of geometric analogues of holographic reduced representations, original holographic reduced representations and binary spatter codes. In *2011 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 221–228. IEEE, 2011.

[39] Peter J Olver and Chehrzad Shakiban. *Applied Linear Algebra.* Springer, 2018.

[40] Manuel Schmuck, Luca Benini, and Abbas Rahimi. Hardware optimizations of dense binary hyperdimensional computing: Rematerialization of hypervectors, binarized bundling, and combinational associative memory. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 15(4):1–25, 2019.

[41] Dmitri Rachkovskij. Linear classifiers based on binary distributed representations. *International Journal Information Theories & Applications*, 2007.

[42] Ernst M Kussul, Lora M Kasatkina, Dmitri A Rachkovskij, and Donald C Wunsch. Application of random threshold neural networks for diagnostics of micro machine tool condition. In *1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98CH36227)*, volume 1, pages 241–244. IEEE, 1998.

[43] EM Kussul. On image texture recognition by associative-projective neurocomputer. In *Proceedings of ANNIE'91 Conference, Intelligent Engineering Systems through Artificial Neural Networks*, pages 453–458. ASME Press, 1991.

[44] DA Rachkovskij and TV Fedoseyeva. On audio signals recognition by multilevel neural network. In *Proceedings of The International Symposium on Neural Networks and Neural Computing-NEURONET*, volume 90, pages 281–283, 1990.

[45] Beth Logan et al. Mel frequency cepstral coefficients for music modeling. In *Ismir*, volume 270, pages 1–11, 2000.

[46] Denis Kleyko and Evgeny Osipov. Brain-like classifier of temporal patterns. In *2014 International Conference on Computer and Information Sciences (ICCOINS)*, pages 1–6. IEEE, 2014.

[47] Mohsen Imani, Samuel Bosch, Sohum Datta, Sharadhi Ramakrishna, Sahand Salamat, Jan M Rabaey, and Tajana Rosing. Quanthd: A quantization framework for hyperdimensional computing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2019.

[48] Abbas Rahimi, Pentti Kanerva, José del R Millán, and Jan M Rabaey. Hyperdimensional computing for noninvasive brain–computer interfaces: Blind and one-shot classification of eeg error-related potentials. In *10th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, number CONF, pages 19–26, 2017.

[49] Ali Moin, Andy Zhou, Abbas Rahimi, Simone Benatti, Alisha Menon, Senam Tamakloe, Jonathan Ting, Natasha Yamamoto, Yasser Khan, Fred Burghardt, et al. An EMG gesture recognition system with flexible high-density sensors and brain-inspired high-dimensional classifier. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE, 2018.

[50] Denis Kleyko, Abbas Rahimi, Dmitri A Rachkovskij, Evgeny Osipov, and Jan M Rabaey. Classification and recall with binary hyperdimensional computing: Tradeoffs in choice of density and mapping characteristics. *IEEE transactions on neural networks and learning systems*, 29(12):5880–5898, 2018.

[51] Mohsen Imani, John Messerly, Fan Wu, Wang Pi, and Tajana Rosing. A binary learning framework for hyperdimensional computing. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 126–131. IEEE, 2019.

[52] Mohsen Imani, Justin Morris, Samuel Bosch, Helen Shu, Giovanni De Micheli, and Tajana Rosing. Adapthd: Adaptive efficient training for brain-inspired hyperdimensional computing. In *2019 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pages 1–4. IEEE, 2019.

[53] Mohsen Imani, Sahand Salamat, Behnam Khaleghi, Mohammad Samragh, Farinaz Koushanfar, and Tajana Rosing. SparseHD: Algorithm-hardware co-optimization for efficient high-dimensional computing. In *2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 190–198. IEEE, 2019.

[54] Tony F Wu, Haitong Li, Ping-Chen Huang, Abbas Rahimi, Gage Hills, Bryce Hodson, William Hwang, Jan M Rabaey, H-S Philip Wong, Max M Shulaker, et al. Hyperdimensional computing exploiting carbon nanotube FETs, resistive RAM, and their monolithic 3D integration. *IEEE Journal of Solid-State Circuits*, 53(11):3183–3196, 2018.

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

[56] Abbas Rahimi, Pentti Kanerva, Luca Benini, and Jan M Rabaey. Efficient biosignal processing using hyperdimensional computing: Network templates for combined learning and classification of ExG signals. *Proceedings of the IEEE*, 107(1):123–143, 2018.

[57] Dmitri A. Rachkovskij. Representation and processing of structures with binary sparse distributed codes. *IEEE Transactions on Knowledge and Data Engineering*, 13(2):261–276, 2001.

[58] Sahand Salamat, Mohsen Imani, Behnam Khaleghi, and Tajana Rosing. F5-hd: Fast flexible FPGA-based framework for refreshing hyperdimensional computing. In *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 53–62, 2019.

[59] Geethan Karunaratne, Manuel Le Gallo, Giovanni Cherubini, Luca Benini, Abbas

Rahimi, and Abu Sebastian. In-memory hyperdimensional computing. *CoRR*, abs/1906.01548, 2019, 1906.01548.

[60] Haitong Li, Tony F Wu, Abbas Rahimi, Kai-Shin Li, Miles Rusch, Chang-Hsien Lin, Juo-Luen Hsu, Mohamed M Sabry, S Burc Eryilmaz, Joon Sohn, et al. Hyperdimensional computing with 3D VRRAM in-memory kernels: Device-architecture co-design for energy-efficient, error-resilient language recognition. In *2016 IEEE International Electron Devices Meeting (IEDM)*, pages 16–1. IEEE, 2016.

[61] UCI machine learning repository. `http://archive.ics.uci.edu/ml/datasets/ISOLET`.

[62] Zisheng Zhang and Keshab K Parhi. Seizure detection using wavelet decomposition of the prediction error signal from a single channel of intra-cranial EEG. In *2014 36th annual international conference of the IEEE engineering in medicine and biology society*, pages 4443–4446. IEEE, 2014.

[63] Zisheng Zhang and Keshab K Parhi. Low-complexity seizure prediction from iEEG/sEEG using spectral power and ratios of spectral power. *IEEE transactions on biomedical circuits and systems*, 10(3):693–706, 2015.

[64] Zisheng Zhang and Keshab K Parhi. Seizure detection using regression tree based feature selection and polynomial SVM classification. In *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 6578–6581. IEEE, 2015.

[65] Zisheng Zhang and Keshab K Parhi. Seizure prediction using polynomial SVM classification. In *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 5748–5751. IEEE, 2015.

[66] Keshab K Parhi and Zisheng Zhang. Discriminative ratio of spectral power and relative power features derived via frequency-domain model ratio with application to seizure prediction. *IEEE transactions on biomedical circuits and systems*, 13(4):645–657, 2019.

[67] Stephen I Gallant and T Wendy Okaywe. Representing objects, relations, and sequences. *Neural computation*, 25(8):2038–2078, 2013.

[68] Hadamard matrix. `https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.linalg.hadamard.html`.

[69] Simon S Haykin. *Adaptive filter theory*. Pearson Education India, 2014.

[70] Pentii Kanerva, Jan Kristoferson, and Anders Holst. Random indexing of text samples for latent semantic analysis. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 22, 2000.

[71] Gabriel Recchia, Magnus Sahlgren, Pentti Kanerva, and Michael N Jones. Encoding sequential information in semantic space models: Comparing holographic reduced representation and random permutation. *Computational intelligence and neuroscience*, 2015, 2015.

[72] Denis Kleyko, Evgeny Osipov, Daswin De Silva, Urban Wiklund, Valeriy Vyatkin, and Damminda Alahakoon. Distributed representation of n-gram statistics for boosting self-organizing maps with hyperdimensional computing. In *International Andrei Ershov Memorial Conference on Perspectives of System Informatics*, pages 64–79. Springer, 2019.

[73] Tharindu Bandaragoda, Daswin De Silva, Denis Kleyko, Evgeny Osipov, Urban Wiklund, and Damminda Alahakoon. Trajectory clustering of road traffic in urban environments using incremental machine learning in combination with hyperdimensional computing. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 1664–1670. IEEE, 2019.

[74] Michael Hersche, Sara Sangalli, Luca Benini, and Abbas Rahimi. Evolvable hyperdimensional computing: Unsupervised regeneration of associative memory to recover faulty components. In *IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS), Genoa, Italy, March 23-25, 2020*. IEEE, 2020.

[75] Denis Kleyko, Evgeny Osipov, Alexander Senior, Asad I Khan, and Yaşar Ahmet Şekercioğğlu. Holographic graph neuron: A bioinspired architecture for pattern processing. *IEEE transactions on neural networks and learning systems*, 28(6):1250–1262, 2016.

[76] Isaac Triguero, Salvador García, and Francisco Herrera. Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. *Knowledge and Information systems*, 42(2):245–284, 2015.

[77] Fateme Rasti Najafabadi, Abbas Rahimi, Pentti Kanerva, and Jan M Rabaey. Hyperdimensional computing for text classification. In *Design, Automation Test in Europe Conference Exhibition (DATE), University Booth*, pages 1–1, 2016.

[78] Abbas Rahimi, Artiom Tchouprina, Pentti Kanerva, José del R Millán, and Jan M Rabaey. Hyperdimensional computing for blind and one-shot classification of EEG error-related potentials. *Mobile Networks and Applications*, pages 1–12, 2017.

[79] Okko Räsänen and Sofoklis Kakouros. Modeling dependencies in multiple parallel data streams with hyperdimensional computing. *IEEE Signal Processing Letters*, 21(7):899–903, 2014.

[80] Okko J Räsänen and Jukka P Saarinen. Sequence prediction with sparse distributed hyperdimensional coding applied to the analysis of mobile phone use patterns. *IEEE transactions on neural networks and learning systems*, 27(9):1878–1889, 2015.

[81] Denis Kleyko, Evgeny Osipov, Ross W Gayler, Asad I Khan, and Adrian G Dyer. Imitation of honey bees' concept learning processes using vector symbolic architectures. *Biologically Inspired Cognitive Architectures*, 14:57–72, 2015.

[82] Ozgur Yilmaz. Connectionist-symbolic machine intelligence using cellular automata based reservoir-hyperdimensional computing. *arXiv preprint arXiv:1503.00851*, 2015.

[83] Denis Kleyko, Sumeer Khan, Evgeny Osipov, and Suet-Peng Yong. Modality classification of medical images with distributed representations based on cellular automata reservoir computing. In *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*, pages 1053–1056. IEEE, 2017.

[84] Guglielmo Montone, J Kevin O'Regan, and Alexander V Terekhov. Hyperdimensional computing for a visual question-answering system that is trainable end-to-end. *arXiv preprint arXiv:1711.10185*, 2017.

[85] Denis Kleyko, Evgeny Osipov, Nikolaos Papakonstantinou, and Valeriy Vyatkin. Hyperdimensional computing in industrial systems: the use-case of distributed fault isolation in a power plant. *IEEE Access*, 6:30766–30777, 2018.

[86] A Mitrokhin, P Sutor, C Fermüller, and Y Aloimonos. Learning sensorimotor control with neuromorphic sensors: Toward hyperdimensional active perception. *Science Robotics*, 4(30):eaaw6736, 2019.

[87] Okko Johannes Räsänen. Generating hyperdimensional distributed representations from continuous-valued multivariate sensory input. In *CogSci*, 2015.

[88] Samuel Bosch, Alexander Sanchez de la Cerda, Mohsen Imani, Tajana Simunic Rosing, and Giovanni De Micheli. QubitHD: A stochastic acceleration method for HD computing-based machine learning. *arXiv preprint arXiv:1911.12446*, 2019.

[89] Mohsen Imani, Yeseong Kim, Sadegh Riazi, John Messerly, Patric Liu, Farinaz Koushanfar, and Tajana Rosing. A framework for collaborative learning in secure high-dimensional space. In *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, pages 435–446. IEEE, 2019.

[90] Zisheng Zhang and Keshab K Parhi. MUSE: Minimum uncertainty and sample elimination based binary feature selection. *IEEE Transactions on Knowledge and Data Engineering*, 31(9):1750–1764, 2018.

[91] Pedro Alonso, Kumar Shridhar, Denis Kleyko, Evgeny Osipov, and Marcus Liwicki. HyperEmbed: Tradeoffs between resources and performance in NLP tasks with hyperdimensional computing enabled embedding of n-gram statistics. *arXiv preprint arXiv:2003.01821*, 2020.

[92] Denis Kleyko, Mansour Kheffache, E Paxon Frady, Urban Wiklund, and Evgeny Osipov. Density encoding enables resource-efficient randomly connected neural networks. *arXiv preprint arXiv:1909.09153*, 2019.

[93] Denis Kleyko, Edward Paxon Frady, and Evgeny Osipov. Integer echo state networks: Hyperdimensional reservoir computing. *arXiv preprint arXiv:1706.00280*, 2017.

[94] Alexander G Anderson and Cory P Berg. The high-dimensional geometry of binary neural networks. *arXiv preprint arXiv:1705.07199*, 2017.

[95] Una Pale, Tomas Teijeiro, and David Atienza. Systematic assessment of hyperdimensional computing for epileptic seizure detection. *arXiv preprint arXiv:2105.00934*, 2021.

[96] Alessio Burrello, Simone Benatti, Kaspar Schindler, Luca Benini, and Abbas Rahimi. An ensemble of hyperdimensional classifiers: Hardware-friendly short-latency seizure detection with automatic ieeg electrode selection. *IEEE journal of biomedical and health informatics*, 25(4):935–946, 2020.

[97] Isabell Kiral-Kornek, Subhrajit Roy, Ewan Nurse, Benjamin Mashford, Philippa Karoly, Thomas Carroll, Daniel Payne, Susmita Saha, Steven Baldassano, Terence O'Brien, et al. Epileptic seizure prediction using big data and deep learning: toward a mobile system. *EBioMedicine*, 27:103–111, 2018.

[98] Nhan Duy Truong, Anh Duy Nguyen, Levin Kuhlmann, Mohammad Reza Bonyadi, Jiawei Yang, Samuel Ippolito, and Omid Kavehei. Convolutional neural networks for seizure prediction using intracranial and scalp electroencephalogram. *Neural Networks*, 105:104–111, 2018.

[99] Benjamin H Brinkmann, Joost Wagenaar, Drew Abbot, Phillip Adkins, Simone C Bosshard, Min Chen, Quang M Tieng, Jialune He, FJ Muñoz-Almaraz, Paloma Botella-Rocamora, et al. Crowdsourcing reproducible seizure forecasting in human and canine epilepsy. *Brain*, 139(6):1713–1722, 2016.

[100] Yun Park, Lan Luo, Keshab K Parhi, and Theoden Netoff. Seizure prediction with spectral power of eeg using cost-sensitive support vector machines. *Epilepsia*, 52(10):1761–1770, 2011.

[101] Florian Mormann, Ralph G Andrzejak, Christian E Elger, and Klaus Lehnertz. Seizure prediction: the long and winding road. *Brain*, 130(2):314–333, 2007.

[102] Josef Parvizi and Sabine Kastner. Human intracranial eeg: promises and limitations. *Nature neuroscience*, 21(4):474, 2018.

[103] John V Guttag, Ali Hossam Shoeb, Blaise Bourgeois, S Ted Treves, Steven C Schachter, Herman A Edwards, John Connolly, et al. Patient-specific seizure onset detection system, October 20 2011. US Patent App. 13/153,819.

[104] Rosana Esteller, Javier Echauz, T Tcheng, Brian Litt, and Benjamin Pless. Line length: an efficient feature for seizure onset detection. In *2001 Conference Proceedings of the 23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, volume 2, pages 1707–1710. IEEE, 2001.

[105] Keshab K Parhi and Zisheng Zhang. Method and apparatus for prediction and detection of seizure activity, October 1 2019. US Patent 10,426,365.

[106] Nhan Duy Truong, Anh Duy Nguyen, Levin Kuhlmann, Mohammad Reza Bonyadi, Jiawei Yang, Samuel Ippolito, and Omid Kavehei. Integer convolutional neural network for seizure detection. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 8(4):849–857, 2018.

[107] Ryan Chen and Keshab K Parhi. Seizure prediction using convolutional neural networks and sequence transformer networks. In *2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pages 6483–6486. IEEE, 2021.

[108] M Dümpelmann. Early seizure detection for closed loop direct neurostimulation devices in epilepsy. *Journal of neural engineering*, 16(4):041001, 2019.

[109] Alessio Burrello, Kaspar Anton Schindler, Luca Benini, and Abbas Rahimi. Hyperdimensional computing with local binary patterns: One-shot learning for seizure onset detection and identification of ictogenic brain regions from short-time ieeg recordings. *IEEE transactions on bio-medical engineering*, 67(2):601–613, 2020.

[110] Steven N Baldassano, Benjamin H Brinkmann, Hoameng Ung, Tyler Blevins, Erin C Conrad, Kent Leyde, Mark J Cook, Ankit N Khambhati, Joost B Wagenaar, Gregory A Worrell, et al. Crowdsourcing seizure detection: algorithm development and validation on human implanted device recordings. *Brain*, 140(6):1680–1691, 2017.

[111] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P Trevino, Jiliang Tang, and Huan Liu. Feature selection: A data perspective. *ACM Computing Surveys (CSUR)*, 50(6):94, 2018.

[112] Una Pale, Tomas Teijeiro, and David Atienza. Exploration of hyperdimensional computing strategies for enhanced learning on epileptic seizure detection. *arXiv preprint arXiv:2201.09759*, 2022.

[113] Han-Tai Shiao, Vladimir Cherkassky, Jieun Lee, Brandon Veber, Edward E Patterson, Benjamin H Brinkmann, and Gregory A Worrell. Svm-based system for prediction of epileptic seizures from ieeg signal. *IEEE Transactions on Biomedical Engineering*, 64(5):1011–1022, 2016.

[114] Farzad Samie, Sebastian Paul, Lars Bauer, and Jorg Henkel. Highly efficient and accurate seizure prediction on constrained IoT devices. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 955–960. IEEE, 2018.

[115] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence*, 27(8):1226–1238, 2005.

[116] Ali Hossam Shoeb. *Application of machine learning to epileptic seizure onset detection and treatment.* PhD thesis, Massachusetts Institute of Technology, 2009.

[117] Mike Heddes, Igor Nunes, Pere Vergés, Dheyay Desai, Tony Givargis, and Alexandru Nicolau. Torchhd: An open-source python library to support hyperdimensional computing research. *arXiv preprint arXiv:2205.09208*, 2022.

[118] American epilepsy society seizure prediction challenge. `https://www.kaggle.com/competitions/seizure-prediction`.

[119] Bhaskar Sen, Shu-Hsien Chu, and Keshab K Parhi. Ranking regions, edges and classifying tasks in functional brain graphs by sub-graph entropy. *Scientific reports*, 9(1):1–20, 2019.

[120] Connectomedb database. `https://db.humanconnectome.org.`

[121] Edward Paxon Frady, Denis Kleyko, and Friedrich T Sommer. Variable binding for sparse distributed representations: Theory and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[122] Pankaj K Agarwal and Nabil H Mustafa. K-means projective clustering. In *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 155–165, 2004.

[123] David Arthur and Sergei Vassilvitskii. K-means++: the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035, 2007.

[124] Stephen C Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

[125] Fionn Murtagh and Pedro Contreras. Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(1):86–97, 2012.

[126] Fionn Murtagh and Pedro Contreras. Algorithms for hierarchical clustering: an overview, II. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 7(6):e1219, 2017.

[127] Lance Parsons, Ehtesham Haque, and Huan Liu. Subspace clustering for high dimensional data: a review. *Acm sigkdd explorations newsletter*, 6(1):90–105, 2004.

[128] Samuel Kaski. Dimensionality reduction by random mapping: Fast similarity computation for clustering. In *1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98CH36227)*, volume 1, pages 413–418. IEEE, 1998.

[129] Saransh Gupta, Behnam Khaleghi, Sahand Salamat, Justin Morris, Ranganathan Ramkumar, Jeffrey Yu, Aniket Tiwari, Jaeyoung Kang, Mohsen Imani, Baris Aksanli, et al. Store-n-learn: Classification and clustering with hyperdimensional

computing across flash hierarchy. *ACM Transactions on Embedded Computing Systems (TECS)*, 21(3):1–25, 2022.

[130] Justin Morris, Kazim Ergun, Behnam Khaleghi, Mohen Imani, Baris Aksanli, and Tajana Simunic. HyDREA: Utilizing hyperdimensional computing for a more robust and efficient machine learning system. *ACM Transactions on Embedded Computing Systems*, 21(6):1–25, 2022.

[131] Brendan J Frey and Delbert Dueck. Clustering by passing messages between data points. *science*, 315(5814):972–976, 2007.

[132] Zhuowen Zou, Hanning Chen, Prathyush Poduval, Yeseong Kim, Mahdi Imani, Elaheh Sadredini, Rosario Cammarota, and Mohsen Imani. BioHD: an efficient genome sequence search platform using hyperdimensional memorization. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*, pages 656–669, 2022.

[133] Linda L Carpenter, Philip G Janicak, Scott T Aaronson, Terrence Boyadjis, David G Brock, Ian A Cook, David L Dunner, Karl Lanocha, H Brent Solvason, and Mark A Demitrack. Transcranial magnetic stimulation (TMS) for major depression: a multisite, naturalistic, observational study of acute treatment outcomes in clinical practice. *Depression and anxiety*, 29(7):587–596, 2012.

[134] Shawn M McClintock, Irving M Reti, Linda L Carpenter, William M McDonald, Marc Dubin, Stephan F Taylor, Ian A Cook, O John, Mustafa M Husain, Christopher Wall, et al. Consensus recommendations for the clinical application of repetitive transcranial magnetic stimulation (rTMS) in the treatment of depression. *The Journal of clinical psychiatry*, 79(1):3651, 2017.

[135] Tyler S Kaster, Jonathan Downar, Fidel Vila-Rodriguez, Kevin E Thorpe, Kfir Feffer, Yoshihiro Noda, Peter Giacobbe, Yuliya Knyahnytska, Sidney H Kennedy, Raymond W Lam, et al. Trajectories of response to dorsolateral prefrontal rtms in major depression: a three-d study. *American Journal of Psychiatry*, 176(5):367–375, 2019.

[136] Cathy S Chen, R Becket Ebitz, Sylvia R Bindas, A David Redish, Benjamin Y Hayden, and Nicola M Grissom. Divergent strategies for learning in males and females. *Current Biology*, 31(1):39–50, 2021.

[137] Claire M Gillan, Michal Kosinski, Robert Whelan, Elizabeth A Phelps, and Nathaniel D Daw. Characterizing a psychiatric symptom dimension related to deficits in goal-directed control. *elife*, 5:e11305, 2016.

[138] Claire M Gillan, Eyal Kalanthroff, Michael Evans, Hilary M Weingarden, Ryan J Jacoby, Marina Gershkovich, Ivar Snorrason, Raphael Campeas, Cynthia Cervoni, Nicholas Charles Crimarco, et al. Comparison of the association between goal-directed planning and self-reported compulsivity vs obsessive-compulsive disorder diagnosis. *JAMA psychiatry*, 77(1):77–85, 2020.

[139] Valerie Voon, Andrea Reiter, Miriam Sebold, and Stephanie Groman. Model-based control in dimensional psychiatry. *Biological psychiatry*, 82(6):391–400, 2017.

[140] Roger Ratcliff and Gail McKoon. The diffusion decision model: theory and data for two-choice decision tasks. *Neural computation*, 20(4):873–922, 2008.

[141] Maxwell Shinn, Norman H Lam, and John D Murray. A flexible framework for simulating and fitting generalized drift-diffusion models. *ELife*, 9:e56938, 2020.

[142] Thomas V Wiecki, Imri Sofer, and Michael J Frank. HDDM: Hierarchical bayesian estimation of the drift-diffusion model in python. *Frontiers in neuroinformatics*, page 14, 2013.

[143] Milton E Strauss, Christopher J McLouth, Deanna M Barch, Cameron S Carter, James M Gold, Steven J Luck, Angus W MacDonald III, J Daniel Ragland, Charan Ranganath, Brian P Keane, et al. Temporal stability and moderating effects of age and sex on cntracs task performance. *Schizophrenia bulletin*, 40(4):835–844, 2014.

[144] Brian M Sweis, Samantha V Abram, Brandy J Schmidt, Kelsey D Seeland, Angus W MacDonald III, Mark J Thomas, and A David Redish. Sensitivity to "sunk costs" in mice, rats, and humans. *Science*, 361(6398):178–181, 2018.

[145] Adam P Steiner and A David Redish. Behavioral and neurophysiological correlates of regret in rat decision-making on a neuroeconomic task. *Nature neuroscience*, 17(7):995–1002, 2014.

[146] Brian M Sweis, Mark J Thomas, and A David Redish. Mice learn to avoid regret. *PLoS Biology*, 16(6):e2005853, 2018.

[147] Samantha V Abram, Yannick-André Breton, Brandy Schmidt, A David Redish, and Angus W MacDonald. The web-surf task: A translational model of human decision-making. *Cognitive, Affective, & Behavioral Neuroscience*, 16:37–50, 2016.

[148] Tao Yu, Yichi Zhang, Zhiru Zhang, and Christopher M De Sa. Understanding hyperdimensional computing for parallel single-pass learning. *Advances in Neural Information Processing Systems*, 35:1157–1169, 2022.

[149] Ruixuan Wang, Xun Jiao, and X Sharon Hu. ODHD: one-class brain-inspired hyperdimensional computing for outlier detection. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pages 43–48, 2022.

[150] scikit-learn. `https://scikit-learn.org`.

[151] Xiao Wang, Fusheng Yu, Witold Pedrycz, and Jiayin Wang. Hierarchical clustering of unequal-length time series with area-based shape distance. *Soft Computing*, 23:6331–6343, 2019.

Appendix A

Supplementary Results

A.1 Clustering Results using HDC.

A.1.1 Small Dataset

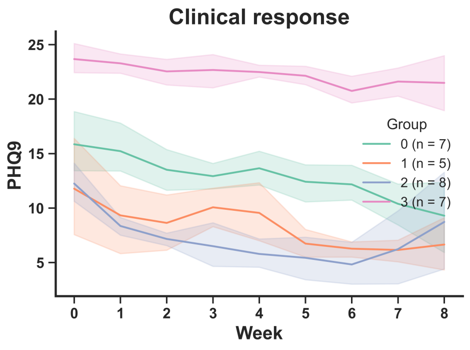
Figure A.1 shows the other six clustering results for the clinical trajectories when $n = 27$. Based on the analysis, it has been recommended that the baseline-corrected data can be grouped into three clusters while the original data can be grouped into four clusters. For binary HDC with record-based encoding, the indication is different. It suggests two clusters for baseline-corrected data and three clusters for original data.

A.1.2 Large Dataset

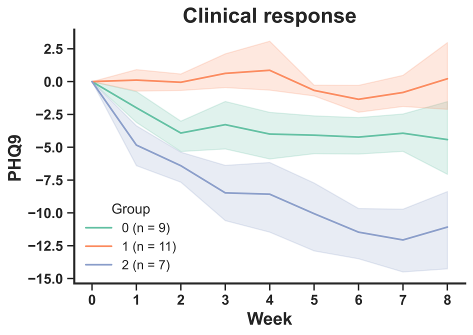
Figure A.2 shows the other six clustering results for the clinical trajectories when $n = 176$. Based on the analysis, it has been recommended that the number of clusters for both original and baseline-corrected data should be equal to or less than seven.

Using N-gram-based encoding for non-binary HDC, original trajectories are grouped into four clusters, and baseline-corrected trajectories are grouped into three clusters when $n = 176$.

Non-binary HDC using record-based encoding

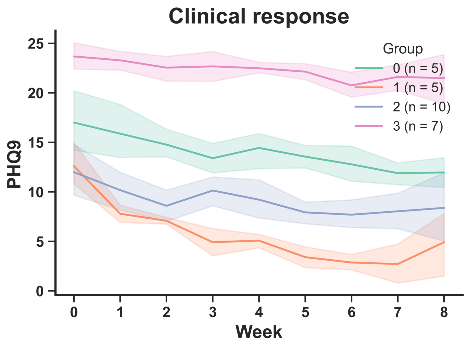


(a) Original data.

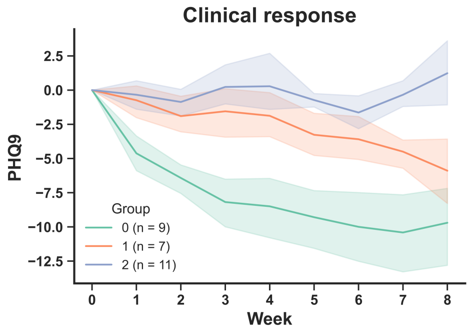


(b) Baseline-corrected data.

Binary HDC using N-gram-based encoding

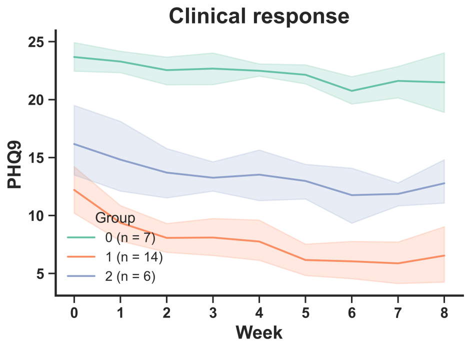


(c) Original data.

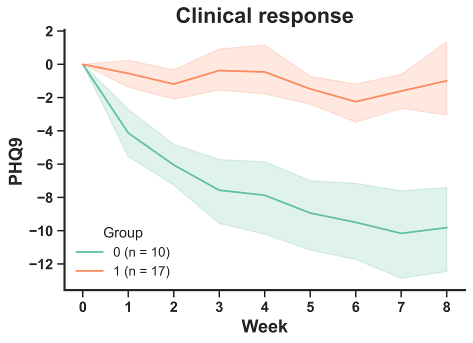


(d) Baseline-corrected data.

Binary HDC using record-based encoding



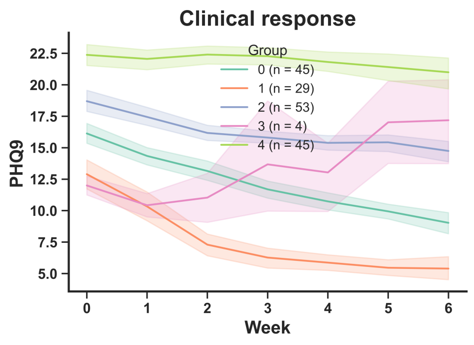
(e) Original data.



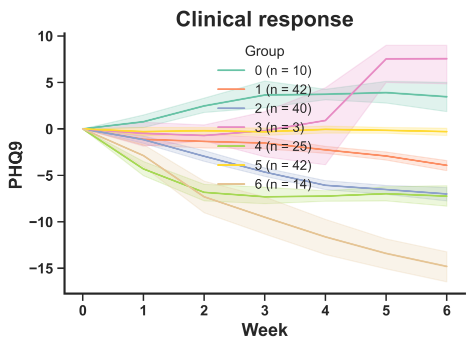
(f) Baseline-corrected data.

Figure A.1: Clinical-trajectory-pattern clustering using HDC for a small dataset ($n = 27$).

Non-binary HDC using record-based encoding

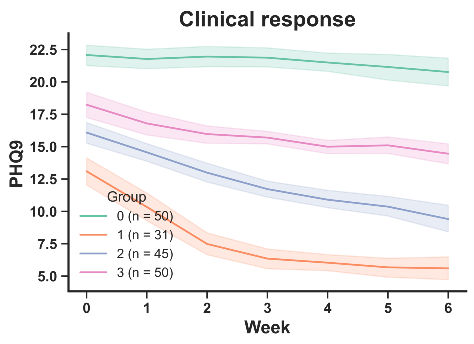


(a) Original data.

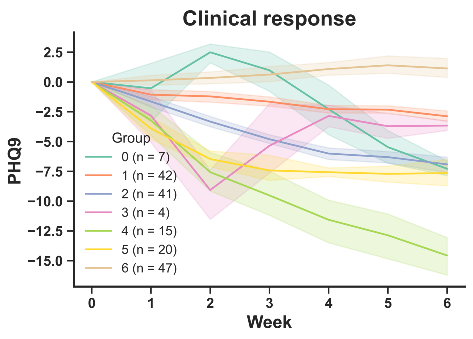


(b) Baseline-corrected data.

Binary HDC using N-gram-based encoding

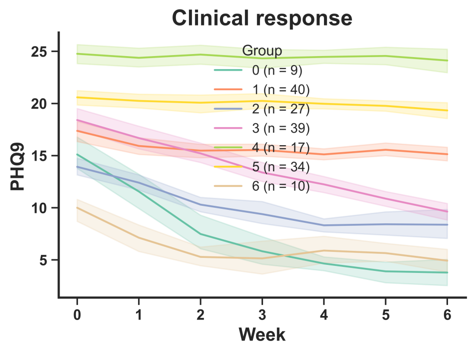


(c) Original data.

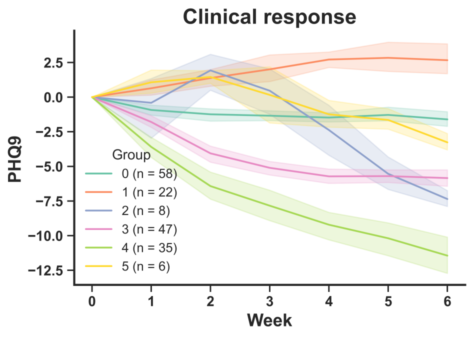


(d) Baseline-corrected data.

Binary HDC using record-based encoding



(e) Original data.



(f) Baseline-corrected data.

Figure A.2: Clinical-trajectory-pattern clustering using HDC for a large dataset ($n = 176$).

A.2 Clustering Results using LCMM.

For longitudinal data, latent class models are typically used.

A.2.1 Large Dataset

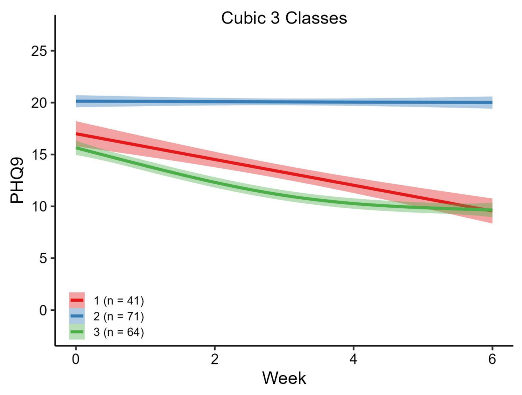
For $n = 176$, Table A.1 summarizes the clustering performance using LCMM for original data, while Table A.2 reports the performance for baseline-corrected data. LCMM indicates the optimal choice when the cubic model is employed. Thus, original data are divided into 3 clusters, whereas baseline-corrected data are divided into 2 clusters. The corresponding clustering patterns are shown in Fig. A.3.

Table A.1: LCMM results for original data ($n = 176$).

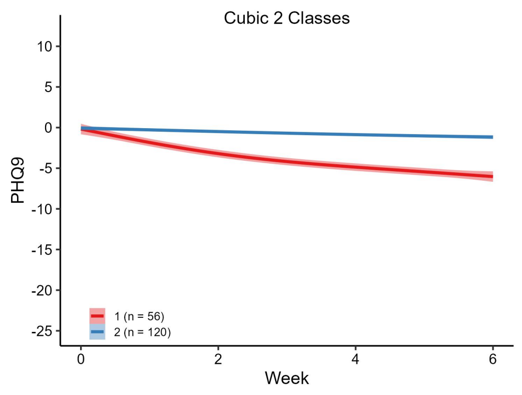
Model	**G**	**AIC**	**Membership [%]**							**MSE**
			group1	**group2**	**group3**	**group4**	**group5**	**group6**	**group7**	
Lin	1	6004.45	100.0							2.49
Lin	2	6004.58	85.80	14.20						2.50
Lin	3	5984.53	53.41	5.68	40.91					2.54
Lin	4	5982.27	53.41	6.82	1.70	38.07				2.57
Lin	5	5987.74	22.16	27.27	35.23	7.39	7.95			2.55
Lin	6	5993.90	33.52	22.16	3.41	6.82	26.70	7.39		2.56
Lin	7	5994.64	22.16	18.18	7.39	12.50	15.91	15.34	8.52	2.57
Quad	1	5828.19	100.0							1.42
Quad	2	5809.75	83.52	16.48						1.42
Quad	3	5810.61	52.84	11.36	35.80					1.43
Quad	4	5809.27	17.61	43.75	3.41	35.23				1.44
Quad	5	5801.60	17.61	42.61	27.27	5.11	7.39			1.44
Quad	6	5795.07	15.91	22.16	27.84	22.16	3.41	8.52		1.47
Quad	7	5774.17	0.57	19.32	18.75	22.16	4.55	27.27	7.39	1.52
Cub	1	5802.02	100.0							1.20
Cub	2	5793.46	4.55	95.45						1.23
Cub	**3**	**5774.26**	**23.30**	**40.34**	**36.36**					**1.11**
Cub	4	5769.66	42.61	18.18	1.70	37.50				1.15
Cub	5	5763.67	50.00	14.77	2.27	10.23	22.73			1.13
Cub	6	5757.28	53.98	6.82	2.84	23.86	2.27	10.23		1.18
Cub	7	5762.22	25.00	16.48	13.07	10.23	23.30	3.98	7.95	1.12

Table A.2: LCMM results for corrected data ($n = 176$).

Model	G	AIC	Membership [%]							MSE
			group1	group2	group3	group4	group5	group6	group7	
Lin	1	5616.39	100.0							2.75
Lin	2	5605.29	99.43	0.57						2.79
Lin	3	5607.15	0.57	81.82	17.61					2.76
Lin	4	5606.00	9.09	36.93	46.02	7.95				2.75
Lin	5	5609.28	9.09	35.80	7.39	14.20	33.52			2.76
Lin	6	5608.08	0.57	39.20	7.39	9.66	40.91	2.27		2.76
Lin	7	5614.29	0.57	7.39	27.84	11.36	39.20	11.36	2.27	2.78
Quad	1	5267.78	100.0							1.44
Quad	2	5246.74	29.55	70.45						1.44
Quad	3	5255.81	15.34	75.57	9.09					1.45
Quad	4	5229.10	0.57	35.80	14.77	48.86				1.46
Quad	5	5230.24	0.57	21.02	24.43	41.48	12.50			1.47
Quad	6	5230.29	0.57	14.20	24.43	19.89	39.77	1.14		1.49
Quad	7	5231.86	0.57	19.32	14.77	25.57	14.20	24.43	1.14	1.49
Cub	1	5195.39	100.0							1.02
Cub	**2**	**5155.13**	**31.82**	**68.18**						**1.00**
Cub	3	5156.13	6.25	34.09	59.66					1.01
Cub	4	5136.23	42.05	18.75	35.23	3.98				1.04
Cub	5	5137.50	3.98	13.07	18.75	36.93	27.27			1.05
Cub	6	5128.61	0.57	10.23	10.80	40.34	35.80	2.27		1.07
Cub	7	5132.27	0.57	21.02	16.48	22.16	24.43	13.07	2.27	1.07



(a) Original data.



(b) Baseline-corrected data.

Figure A.3: Trajectory clustering for LCMM when $n = 176$.