Neurally-Inspired Hyperdimensional Classification for Efficient and Robust Biosignal Processing

Yang Ni[1], Nicholas Lesica[2], Fan-Gang Zeng[1], and Mohsen Imani[1*]
[1]University of California Irvine, [2]University College London
[*]Corresponding Author: m.imani@uci.edu

ABSTRACT

The biosignals consist of several sensors that collect time series information. Since time series contain temporal dependencies, they are difficult to process by existing machine learning algorithms. Hyper-Dimensional Computing (HDC) is introduced as a brain-inspired paradigm for lightweight time series classification. However, there are the following drawbacks with existing HDC algorithms: (1) low classification accuracy that comes from linear hyperdimensional representation, (2) lack of real-time learning support due to costly and non-hardware friendly operations, and (3) unable to build up a strong model from partially labeled data.

In this paper, we propose TempHD, a novel hyperdimensional computing method for efficient and accurate biosignal classification. We first develop a novel non-linear hyperdimensional encoding that maps data points into high-dimensional space. Unlike existing HDC solutions that use costly mathematics for encoding, TempHD preserves spatial-temporal information of data in original space before mapping data into high-dimensional space. To obtain the most informative representation, our encoding method considers the non-linear interactions between both spatial sensors and temporally sampled data. Our evaluation shows that TempHD provides higher classification accuracy, significantly higher computation efficiency, and, more importantly, the capability to learn from partially labeled data. We evaluate TempHD effectiveness on noisy EEG data used for a brain-machine interface. Our results show that TempHD achieves, on average, 2.3% higher classification accuracy as well as 7.7× and 21.8× speedup for training and testing time compared to state-of-the-art HDC algorithms, respectively.

ICCAD '22, October 30-November 3, 2022, San Diego, CA, USA
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9217-4/22/10.
https://doi.org/10.1145/3508352.3549477

ACM Reference Format:
Yang Ni[1], Nicholas Lesica[2], Fan-Gang Zeng[1], and Mohsen Imani[1*]. 2022. Neurally-Inspired Hyperdimensional Classification for Efficient and Robust Biosignal Processing . In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD '22), October 30-November 3, 2022, San Diego, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3508352.3549477

1 INTRODUCTION

In many real-world applications, data are captured over the course of time, constituting a time series. Particularly, biosignals consist of several sensors that collect time series information [1]. Since time series contain temporal dependencies, it is often difficult to analyze them using machine learning algorithms. Deep learning algorithms have been commonly used to extract incident patterns or insights from data and build a supervised learning model. However, the current deep learning models have a weak notion of memorization, while biosignals often deal with temporal information. Recent research tried to use recurrent neural networks (RNN) and Long short-term memory (LSTM) to learn and memorize spatial-temporal information [2, 3]. However, these models are significantly difficult and inefficient to train. In addition, running deep learning algorithms requires large off-chip memory to store train data and perform many learning iterations [4]. This resource requirement is often above the capability of today's embedded devices. Lastly, deep learning solutions require many train data to build up a reliable model, while in practice, it is costly and sometimes impossible to collect such labeled data.

Hyper-Dimensional Computing (HDC) is introduced as an alternative paradigm that mimics important brain functionalities towards high-efficiency and noise-tolerant computation [4–8]. HDC is motivated by the observation that the human brain operates on high-dimensional data representations. In HDC, objects are thereby encoded with high-dimensional vectors, called *hypervectors*, which have thousands of elements [5]. HDC incorporates learning capability along with typical memory functions of storing/loading information. This makes HDC capable of dealing with noisy time-series data. In addition, HDC mimics important functionalities of the human memory model with vector operations, which are computationally tractable and mathematically rigorous in describing human cognition.

Several recent works used HDC as a lightweight solution for biosignal classification. For example, works in [9–13] used HDC for real-time classification from Electroencephalography (EEG) and Electromyography (EMG) sensors. A key advantage is HDC training capability in one or a few shots, where data can be learned with few iterations. Research above already achieved comparable accuracy with fewer learning iterations as compared to support vector machines (SVMs), gradient boosting, and neural networks. Despite the success, there are three major drawbacks with the existing HDC solutions: (1) HDC algorithms use non-flexible mathematics, which has difficulty in encoding and representing complex time-series data. This results in a lower learning accuracy over complex time-series tasks. (2) The current HDC encoding methods use costly high-dimensional operations to preserve spatial-temporal relations. These operations are not friendly with existing embedded processors. For example, existing HDC methods use permutation (rotational shift) operation to preserve the orders in a sequence. Implementing this operation in the current CPU/GPU requires thousands of read/write operations in memory. (3) The existing HDC algorithms are sensitive to the amount of labeled data provided to learn a model. This is an essential factor for biosignal processing, as collecting labeled data is often very costly or sometimes impossible.

In this paper, we propose TempHD, a novel hyperdimensional computing method for efficient and accurate biosignal classification. Our solution provides a better representation of data in high-dimensional space with higher separability, thus providing a higher learning capability and classification accuracy with fewer labeled samples. The main contributions of the paper are as follows:

- In contrast to the existing HDC algorithms that linearly map data points into the hyperspace, we develop a non-linear encoding, inspired by the Kernel method, that maps spatial information into high-dimensional space. The proposed encoding explicitly considers the non-linear interactions between the input features. This enables TempHD to have a more informative representation, thus providing easier data separation in high-dimension.
- We expand the proposed non-linear encoder to also preserve temporal information. Instead of using costly high-dimensional mathematics, TempHD represents temporal information by sorting original data before feeding them into kernel encoding. This eliminates thousands of operations that are currently required to implement a simple ordering of signals in high-dimension. Our solution also considers the interactions between the correlated sampled signals, thus reducing sensitivity to the sampling noise.
- We develop learning algorithms that adaptively update HDC model to eliminate possible model saturation. During training, we update HDC model adaptively depending on the already stored information in the model. This results in a higher learning accuracy.

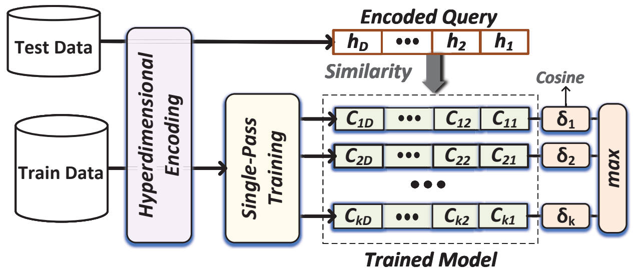

Figure 1: Hyperdimensional classification during both training and inference.

We evaluate TempHD effectiveness on noisy EEG data used for brain-machine interface [14]. Our evaluation shows that TempHD provides higher classification accuracy, significantly higher computation efficiency, and, more importantly, the capability to learn from partially labeled data. Our results indicate that TempHD achieves, on average, 2.3% higher classification accuracy as well as 7.7× and 21.8× speedup for training and testing time compared to state-of-the-art HDC algorithms, respectively.

2 HYPERDIMENSIONAL COMPUTING

Hyper-Dimensional computing (HDC) is based on the understanding that brains compute with patterns of neural activity that are not readily associated with numbers. Due to the very size of the brain's circuits, neural patterns can be modeled with hypervectors [5]. HDC builds upon a well-defined set of operations with random hypervectors, is extremely robust in the presence of failures, and offers a complete computational paradigm that is easily applied to learning problems. There exist a huge number of different, nearly orthogonal hypervectors with dimensionality in the thousands.

Figure 1 shows an overview of HDC learning. The first step in HDC is to encode data into high-dimensional space. Then, HDC performs a learning task over encoder data by performing a single-pass training. The result of training will be to generate a hypervector representing each class. The inference task can be performed by checking the similarity of an encoded query to the class hypervector.

2.1 HDC Primitives

The encoder is the most important component of hyperdimensional learning. The goal of the encoder is to map data into high-dimensional space such that we can extract knowledge from data at a lower cost. The HDC encoding is performed based on a well-defined set of mathematics. Let us assume $\vec{\mathcal{H}}_1, \vec{\mathcal{H}}_2$ are two randomly generated hypervectors, where $\vec{\mathcal{H}} \in \{-1, +1\}^D$ and $\delta(\vec{\mathcal{H}}_1, \vec{\mathcal{H}}_2) \approx 0$. D is the dimension of the HDC space. HDC works based on a set of primitives: *(1) Bundling:* is an addition of multiple hypervectors into a single hypervector, $\vec{\mathcal{R}} = \vec{\mathcal{H}}_1 + \vec{\mathcal{H}}_2$. Unlike original space that bundling act as an average operation, in high-dimensional space the addition is memorization function. *(2) Binding:* associates multiple orthogonal hypervectors (e.g., $\vec{\mathcal{H}}_1$, $\vec{\mathcal{H}}_2$) into a single hypervector ($\vec{\mathcal{R}} = \vec{\mathcal{H}}_1 * \vec{\mathcal{H}}_2$). The binded hypervector is a new object in HDC space which is orthogonal to all input hypervectors ($\delta(\vec{\mathcal{R}}, \vec{\mathcal{H}}_1) \simeq 0$ and $\delta(\vec{\mathcal{R}}, \vec{\mathcal{H}}_2) \simeq 0$). *(3) Permutation:* defines as a single rotational shift. The permuted hypervector will be nearly orthogonal to its original hypervector ($\delta(\vec{\mathcal{H}}_1 \rho \vec{\mathcal{H}}_1) \simeq 0$). (4) **Reasoning** is done by measuring the similarity of hypervectors. We denote the cosine similarity with $\delta(\vec{\mathcal{H}}_1, \vec{\mathcal{H}}_2)$.

2.2 Time-series Encoding

As Figure 2a shows, the time series often get sampled in an n-gram window. In each sampling window, the signal values (in the y-axis) store the information, and the time (x-axis) represents the sequence. We assign a random vector to V_{min} ($\vec{L}_{min}$ representing minimum signal value) and V_{max} ($\vec{L}_{max}$ representing maximum signal value). Since these vectors are randomly generated, they are nearly orthogonal. For signal values between V_{min} and V_{max}, we perform vector quantization to generate vectors that have a spectrum of similarity to $\vec{L}_{min}$ and $\vec{L}_{max}$. Finally, the encoding can be performed by binding the level hypervectors corresponding to sampled signal while using permutation to store the timing information. For example shown in Figure 2a, the trigram Windows can be encoded as $\vec{\mathcal{H}} = \rho\rho\vec{L}_{t_3} * \rho\vec{L}_{t_2} * \vec{L}_{t_1}$. Note that for time-series, the encoding repeats after moving a sliding window one time-step ahead. This will generate a large number of encoded data that can be used for training.

For applications with multi-sensors, the encoding follows the same procedure for each sensor using synchronized n-gram windows. Let us assume a problem with m sensors, where sensors generate the following encoded hypervectors $\{\vec{\mathcal{H}}_1, \vec{\mathcal{H}}_2, \cdots, \vec{\mathcal{H}}_m\}$. We accordingly generate m random hypervector, where each is a signature of a sensor $\{\vec{\mathcal{P}}_1, \vec{\mathcal{P}}_2, \cdots, \vec{\mathcal{P}}_m\}$. To aggregate information, the encoded hypervector from each sensor will be binded (associated) with the corresponding identification hypervector:

$$\vec{\mathcal{H}} = \vec{\mathcal{P}}_1 * \vec{\mathcal{H}}_1 + \vec{\mathcal{P}}_2 * \vec{\mathcal{H}}_2 + \cdots + \vec{\mathcal{P}}_m * \vec{\mathcal{H}}_m \qquad (1)$$

where '+' memorizes the sensor information and each $\vec{\mathcal{P}}$ preserves the position of a sensor.

2.3 Challenges

Despite the effectiveness of the above encoding and learning methods, There is still three main drawbacks that are listed below:

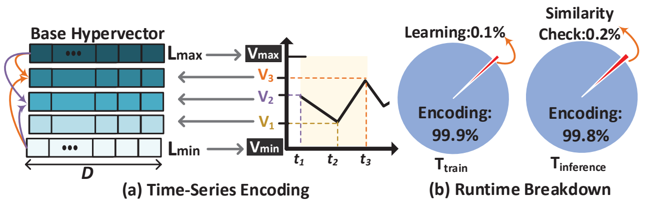

Figure 2: (a) Existing time-series encoding and (b) Breakdown of HDC execution time during training and inference.

- The existing encoding methods linearly combine the hypervectors corresponding to each feature, resulting in suboptimal classification quality for general and complex classification problems. To obtain the most informative hypervectors, the HDC encoding should consider the non-linear interactions between the spatial sensors and temporally sampled data.
- The encoding method is associated with high computational cost. For example, the permutation used to preserve the temporal correlation is not hardware friendly as it requires thousands of read/write operations in memory. Figure 2b shows the breakdown of HDC execution time. Our evaluation shows that the encoding module takes 99.9% and 99.8% of total execution time during training and inference. The results are reported when classifying EEG signals running state-of-the-art HDC algorithms on the CPU. The details of the experimental platform and dataset are listed in Section 5.
- The current HDC models are weak in learning from partially labeled data. However, in practice, it is hard or impossible to collect labeled data from biosensors. Therefore, it is essential to develop algorithms that can effectively learn from few labeled data.

3 KERNEL-BASED ENCODING

In this section, we introduce our encoding method that exploits the kernel trick [15, 16] to map data points into the high-dimensional space. The underlying idea of the kernel trick is that data, which is not linearly separable in original dimensions, might be linearly separable in higher dimensions. Let us consider certain functions $K(x, y)$ which are equivalent to the dot product in a different space, such that $K(x, y) = \Phi(x) \cdot \Phi(y)$, where $\Phi(\cdot)$ is often a function for high dimensional projection. We can take advantage of this implicit mapping by replacing their decision function with a weighted sum of kernels:

$$f(\cdot) = \sum_{i=0}^{N} c_i K(\cdot, x_i) \qquad (2)$$

where (x_i, y_i) is the training data sample, and the c_is are constant weights. The study in [15] showed that the inner

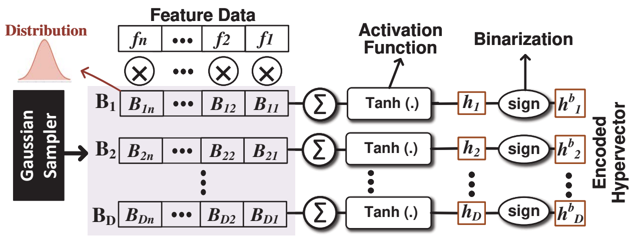

Figure 3: Hyperdimensional kernel-based encoder.

product can efficiently approximate Radial Basis Function (RBF) kernel, such that:

$$K(x, y) = \Phi(x) \cdot \Phi(y) \approx z(x) \cdot z(y) \tag{3}$$

The Gaussian kernel function can now be approximated by the dot product of two vectors, $z(x)$ and $z(y)$.

The proposed encoding method is inspired by the RBF kernel trick method. Figure 3 shows our encoding procedure. Assume an input vector in original space $\vec{F} = \{f_1, f_2, \cdots, f_n\}$ and $\vec{F} \in \mathcal{R}^n$. The encoding module maps this vector into high-dimensional vector, $\vec{\mathcal{H}} = \{h_1, h_2, \cdots, h_D\} \in \mathcal{R}^D$, where $D \gg n$. The following equation shows an encoding method that maps input vector into high-dimensional space:

$$h_i = \cos(\vec{F} \cdot \vec{\mathcal{B}}_i + b_i)\sin(\vec{F} \cdot \vec{\mathcal{B}}_i) \tag{4}$$

where $\vec{\mathcal{B}}_k$s are randomly chosen hence orthogonal base hypervectors of dimension $D \simeq 10k$ to retain the spatial or temporal location of features in an input and $b_i \sim \mathcal{U}(0, 2\pi)$. That is, $\vec{\mathcal{B}}_{kj} \sim \mathcal{N}(0, 1)$ and $\delta(\vec{\mathcal{B}}_{k_1}, \vec{\mathcal{B}}_{k_2}) \simeq 0$, where δ denotes the cosine similarity. However, this activation is not a convex function, thus making it impossible to back-propagate from HDC encoder. To enable HDC model to support back-propagation, one can decide to exploit hyperbolic tangent function as an activation function: $h_i = \tanh(\vec{F} \cdot \vec{\mathcal{B}}_i + b_i)$.

3.1 Non-Linear Time-Series Encoding

The above encoding is explained in the context of feature vectors to preserve spatial correlation. However, time series encoding also needs to preserve temporal information. Here, we show how the proposed kernel-based encoding can be extended to support temporal correlation. Let us consider the time series data shown in Figure 4a. We first sample data in an n-gram window. The window size depends on the nature of the signal and sampling time; however, it is often smaller than the time-series length. Regardless of windows size, the goal of HDC encoding is to represent the pattern of a sampled signal as a high-dimensional pattern. In this representation, we need to store sampled Y values along with the corresponding time that they occur. In the example shown in Figure 4b, we sort the sampled signal value depending on their sampling time. In our example, $\{Y_1, Y_2, \cdots, Y_n\}$ are happening in $\{t_1, t_2, \cdots, t_n\}$. As Figure 4b shows, this temporally sorted data can be treated as a feature vector; thus, it can be mapped using our proposed kernel-based encoder (explained in Section 3).

To encode the entire time-series, the n-gram windows will move over the signal in an overlapping manner. Our encoding repeats the same process by sorting the sampled signal values and feeding them into our non-linear encoder. This process continues until covering the entire time-series data. In the case of using a single sensor, the encoded data can be sent to the classification module for training. However, in practice, many biosignals often have multiple sensor data. For example, conventional EEG sensors consist of 64 electrodes collecting information of different brain regions [17, 18]. Similarly, multi-electrode EMG signals are required for applications such as gesture detection [9]. Our encoding maps the multi-channel encoding by spatially sorting samples obtained from different sensors. Let us assume an example with m sensors; all sampled synchronously over n-gram windows. To consider the location of each sensor, our solution not only sorts the sampled signal from each sensor but also sorts the sample based on the index of the sensor. For example, the samples coming from sensor 1 will locate in indices $[1 : n]$, while the k^{th} sensor samples will locate in $[n \times (k-1)+1 : n \times k]$ indices. This spatially sorted sensor signals will be a vector with $n \times m$ length and can be sent to our proposed kernel encoder for projecting into high-dimensional space (Figure 4c).

This encoding provides multiple advantages as compared to existing hyperdimensional encoding for time-series encoding:

(1) Quality of Encoding: Each row of the kernel encoder is a projection vector that adds up spatial-temporal information of all sensors with different weights (Figure 4d). Since the weights are non-binary, the encoder considers the non-linear interactions between the sensors and their temporal samples. In addition, the activation function used after the mapping mathematically gives non-linearity to our encoder to better represent data in high-dimension. This is an important factor since the sensors, e.g., electrodes in EEG sensors, are not independent. For example, physically close electrodes are likely to read correlated data. However, due to the complexity of the systems and sensors, it is hard to learn the relation. Instead, our encoder considers non-linear interactions that possibly occur between them. Note that the existing encoding modules linearly add up the sensor data by associating each sensor with a random bipolar vector. As a result, they have a more linear nature in both temporal and spatial mapping. In Section 5.2, we show how our encoding outperforms existing HDC algorithms in terms of accuracy.

Besides the accuracy, our non-linear data encoding enables better data separation in high-dimensional space. Therefore, our learning models can provide high classification accuracy

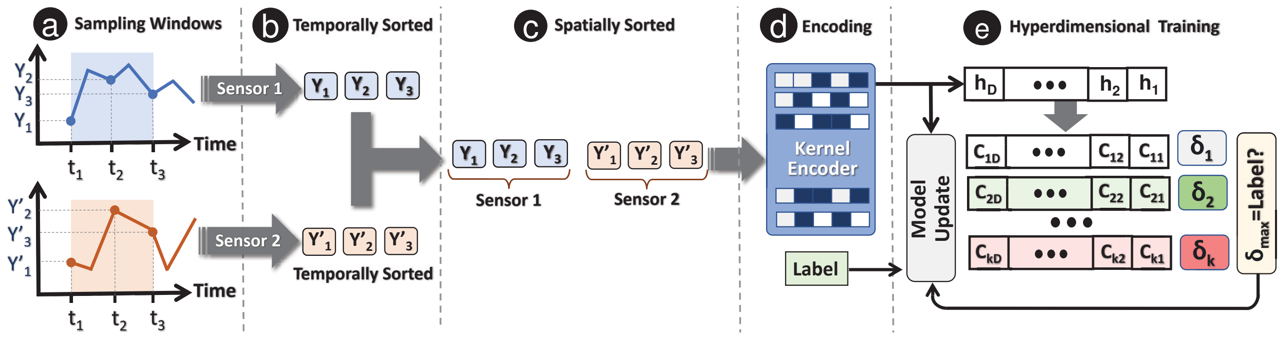

Figure 4: (a-c) Overview of our spatial-temporal encoder, (d) kernel encoder to map the sorted features into high-dimension (e) hyperdimensional classification.

with very few labeled samples. In Section 5.6, we show HDC capability in learning from partially labeled data.

(2) Computation Efficiency: Existing HDC methods map sampled data into high-dimensional space in a very initial step of an encoding process [9, 10, 13]. The sampled signals in each time series translate to a hypervector with thousands of dimensions. As a result, both temporal and spatial information needs to be preserved by computing over hypervectors. This means that they require thousands of operations for each encoding operation. Unfortunately, these operations are inefficient existing hardware platforms, such as CPU and GPU. Similar cost associates with spatial encoding, where the position of each sensor is preserved by associating it with a random hypervector. This also involves thousands of computations to ensure a simple and pre-defined sort operation. In contrast, our solution provides a new perspective to develop efficient HDC encoding modules. Our solution ensures both spatial and temporal information to preserve before mapping data into high-dimensional space. As a result, we have much higher parallelism and lower computational costs, which are essential for learning on embedded devices. In Section 5.3, we compare TempHD efficiency with state-of-the-art learning solutions for biosignal classification.

4 HYPERDIMENSIONAL MODEL TRAINING

We exploit hyperdimensional learning to directly operate over encoded data. Figure 4e shows an overview of HDC classification. HDC identifies common patterns during learning and eliminates the saturation of the class hypervectors during single-pass training. Instead of naively combining all encoded data, our approach adds each encoded data to class hypervectors depending on how much new information the pattern adds to class hypervectors. If a data point already exists in a class hypervector, HDC will add no or a tiny portion of data to the model to prevent hypervector saturation. If the prediction matches the expected output, no update will be made to avoid overfitting. This adaptive update provides a higher chance and weight to non-common patterns to represent the final model. This method can eliminate the necessity of using costly iterative training.

Let us assume $\vec{\mathcal{H}}$ as a new training data point. HDC computes the cosine similarity of $\vec{\mathcal{H}}$ with all class hypervectors $\vec{C}$s. We compute similarity of this data point with class i as: $\delta_i = \delta(\vec{\mathcal{H}}, \vec{C}_i)$. Instead of naively adding a data point to the model, HDC updates the model based on the δ similarity. If an input data has label l and correctly matches with the class, the model updates as follows:

$$\vec{C}_l \leftarrow \vec{C}_l + \eta_1\ (1 - \delta_l) \times \vec{\mathcal{H}} \tag{5}$$

where η is a learning rate. A large δ_l indicates that the input is a common data point which is already exist in the model. Therefore, our update adds a very small portion of encoded query to model to eliminate model saturation ($1 - \delta_l \simeq 0$). If the input data get an incorrect label of l', the model updates as:

$$\vec{C}_{l'} \leftarrow \vec{C}_{l'} - \eta_2\ (\delta_{l'} - \delta_l) \times \vec{\mathcal{H}} \tag{6}$$

where $\delta_{l'} - \delta_l$ determines the weight that the model needs to be updated. Small $\delta_{l'} - \delta_l$ indicates that the query is marginally mismatched while larger mismatch is updated with a larger factor ($\delta_{l'} - \delta_l \gg 0$).

4.1 Hyperdimensional Inference

In inference, HDC checks the similarity of each encoded test data with the class hypervector in two steps. The first step encodes the input (the same encoding used for training) to produce a query hypervector $\vec{\mathcal{H}}$. Then we compute the similarity (δ) of $\vec{\mathcal{H}}$ and all class hypervectors. Query data gets the label of the class with the highest similarity.

5 EVALUATION

5.1 Experimental Setup

Our method is general and applicable to classify a wide range of biosignals, including multi-channel EMG and EEG signals. In this work, we look at a particular application of TempHD in the noninvasive Brain-Computer Interfaces (BCI). One of the targets in BCI is to recognize user intention from collected biosignals, thereby enables controlling without body movement. For example, in [1, 19, 20], they request the human user to move a cursor to a specific location using mental commands. However, direct prediction for human intention using noninvasive biosignals often leads to low accuracy. Thus, researchers approach this problem by using error-related biosignals, e.g., EEG error-related potentials (ERP). Instead of being directly controlled by BCI and recognized human intention, the computer operates and learns by itself. In addition, it also tries to recognize the human response to its behavior via ERP, i.e., whether the human considers its behavior as correct or not. Our TempHD is particularly designed to classify biosignal with high efficiency and lower inference delay, so it is suitable for ERP recognition in BCI control tasks.

As the workload for our design, we use ERP signals collected using EEG sensors [14]. In this particular dataset, EEG signals are collected from six subjects using 64 electrodes. The task for subjects is moving a cursor to a target location, which is similar to the example we mentioned before. However, they have no control over the computer, and the cursor is moving by itself either towards the correct direction or the wrong one. For each trial, the probability of the cursor moving in the wrong direction is 0.2. Subjects are requested to judge the correctness of direction, and researchers collect the corresponding EEG signals of six subjects. The trial is labeled according to the correctness of the cursor direction. We use TempHD for direct ERP classification by learning on the EEG 64-channel raw waveform. We only apply simple data preprocessing, which is a band-pass filter of 1-10 Hz for noise reduction. The waveform length of each trial is around 2000 ms with the sampling rate of 512 Hz; and the effective time window of each trial is up to 600 ms. We use downsampling to reduce the number of data points in each time window for efficient HDC learning. The parameters of this dataset are shown in Table 1.

During the experiment on this dataset, we test our TempHD by comparing the accuracy and efficiency with SVM and previous HDC method for time-series data, which is described in Section 2.2. We also compare the performance of two HDC methods under different dimensionality settings. By varying the size of the training dataset, we present the different learning accuracy of each method. The default dimensionality for both HDC methods is 10000, except for Section 5.4 where the dimensionality changes from 500 to 10000. The default number of iterations for TempHD training is 300. We collect all the results on the CPU platform with AMD Ryzen-5 3600X.

Table 1: Time window & data points per channel for subjects.

Subject	Time window (ms)	Data points per window	Downsampling rate (points/step)	Data points after downsampling
S1	250	128	8	16
S2	450	230	8	29
S3	250	128	8	16
S4	600	307	16	19
S5	450	230	8	29
S6	450	230	8	29

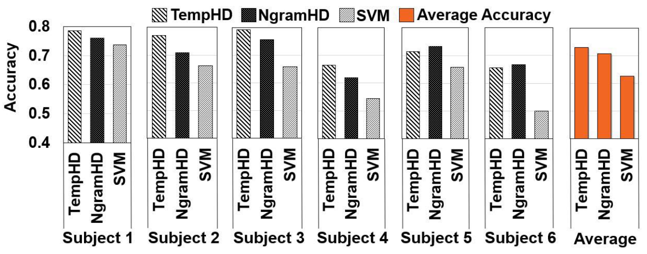

Figure 5: Single-subject accuracy and multi-subject average accuracy comparison between TempHD, **NgramHD, and SVM**

5.2 TempHD Accuracy

In Figure 5, we show the classification accuracy for each subject and the average accuracy over all six subjects. We compare the results of our TempHD with two methods: i) state-of-the-art HDC classification (NgramHD) [9, 21], and ii) support vector machine (SVM). TempHD with non-linear encoding achieves better accuracy in subjects 1 to 4 and also multi-subject average. Our TempHD gets around 80% accuracy for subject 1 and subject 3; it also achieves 73% accuracy, on average, which is about 3% higher than the NgramHD method's accuracy. In addition, TempHD significantly improves the accuracy compared to SVM. For example, TempHD accuracy is 15.5% higher in subject 6 and 10% higher on average.

5.3 TempHD Efficiency

As shown in Figure 6, we collect the runtime results for training and testing on each subject for TempHD, NgramHD, and SVM. Although the accuracy results for SVM are the lowest, it achieves the fastest training and testing among the three methods because the dataset sizes are relatively small. However, SVM is known to require a long training time for large datasets, and biosignal datasets may include a large amount of data for practical applications. Our TempHD, on

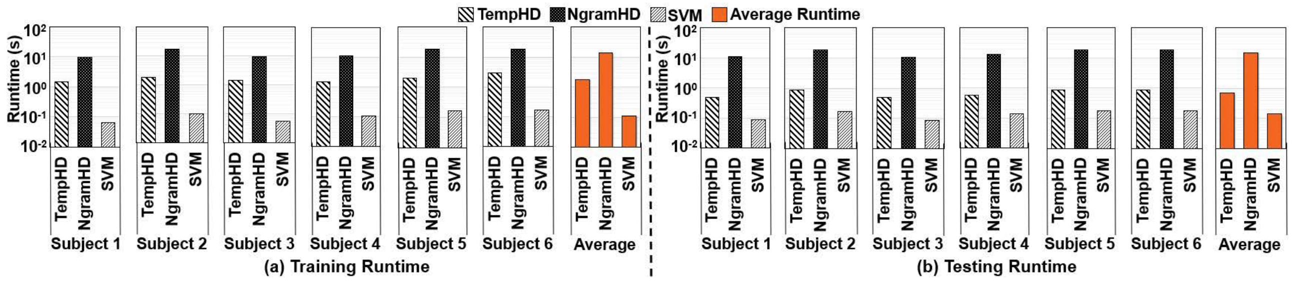

Figure 6: Single-subject and multi-subject average runtime comparison between TempHD, NgramHD, and SVM.

the other hand, not only achieves the highest accuracy but also significantly improves the runtime cost comparing to the previous HDC method. The speedup of TempHD over NgramHD is around 7.7× for average training runtime; and for average testing runtime, the speedup is about 21.8×. The results show that our TempHD is more suitable and more efficient for biosignal and multi-channel time series classification.

We also explore the performance scalability of SVM and our TempHD by increasing the dataset size so that we can better infer the performance of both methods with larger biosignal datasets. We pick the training and testing dataset for subject 1 and expand it with different ratios from about 5× to 200×. The original training and testing dataset size are 489 and 555, respectively. We present the results in Table 2, which records the training and testing runtime for different dataset sizes. We observe that the training time of SVM grows much faster than that of TempHD. SVM is faster in training when the dataset size is small; however, it is nearly 8× slower than TempHD when the number of training samples is 100k. In fact, SVM also has a larger testing time when we increase the dataset size, i.e., about 2× slower for testing when we expand the testing samples to 112k. Both results align with the fact that SVM is slow for large datasets. On the other hand, our TempHD scales better with increasing dataset size.

5.4 TempHD & Dimensionality

In this section, we explore the effect of dimensionality on both classification accuracy and runtime. As shown in Figure 7, we compare the performance and efficiency of our TempHD and NgramHD under different dimensions. The accuracy and runtime results are averaged over six subjects. When increasing the dimensionality from 500 to 10,000, we observe that the accuracy and runtime for both methods increase as expected. However, TempHD constantly achieves better accuracy and has significantly lower runtime cost for all dimension settings. Notice that in Figure 7(a), our TempHD achieves over 70% accuracy with dimensions lower than 1000. This improvement over the previous HDC method is crucial because lower dimensionality leads to lower computation and runtime cost, especially for low-power devices. In

Table 2: TempHD performance scalability with data size.

# Train Samples		2.5k	5k	12.5k	25k	50k	75k	100k
Training	*SVM*	0.5s	1.2s	3.1s	6.1s	12.2s	451.6s	808.0s
	TempHD	3.0s	5.7s	14.0s	27.6s	55.1s	82.0s	111.3s
# Test Samples		**2.8k**	**5.6k**	**14k**	**28k**	**56k**	**84k**	**112k**
Testing	*SVM*	0.6s	1.4s	3.3s	6.8s	13.7s	20.0s	26.5s
	TempHD	0.5s	0.8s	2.0s	3.4s	6.8s	10.1s	13.7s

Figure 7(b) and (c), we observe that TempHD achieves nearly 40× speedup in training runtime and about 741× speedup in testing with 500 dimensions.

5.5 TempHD Breakdown

In Figure 8, we present the breakdown pie chart for the execution time of training and testing. The encoding time of TempHD still dominates in both the training and testing process, i.e., 65.1% for training set encoding and 99.6% for testing set encoding. Unlike the learning process in NgramHD, which takes up only 0.1% of the total training time, the adaptive multi-iteration learning in TempHD composes 34.9%. Although iterative learning requires more execution time, the fast encoding of TempHD significantly reduces the overall training time. Also, our iterative learning with an adaptive learning rate is the key to higher accuracy.

5.6 TempHD with Partial Train Data

In Figure 9, we observe the change of average classification accuracy with varying percentages of training trials. We randomly sample a portion of training samples from the training dataset with the percentage changing from 10% to 90%. Notice that our TempHD provides nearly 60% accuracy with only 10% of the training samples. This is an appealing characteristic because the number of training samples is usually limited in practical applications. The figure also shows that TempHD constantly achieves better accuracy than NgramHD and significantly improves the classification quality comparing to the SVM method.

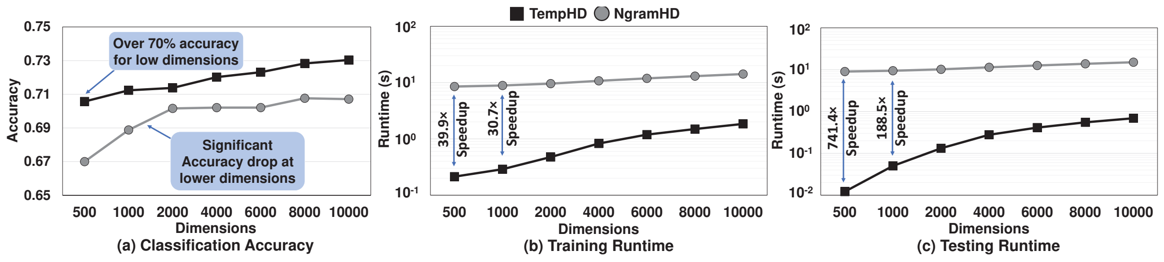

Figure 7: Average accuracy and runtime comparison with varying dimensionality from 500 to 10,000

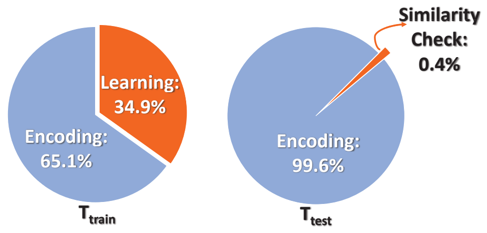

Figure 8: TempHD execution Breakdown.

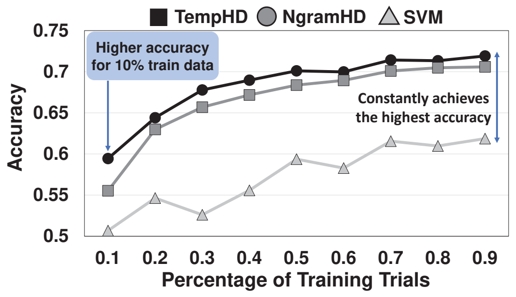

Figure 9: accuracy comparison during partial training.

6 RELATED WORKS

Prior research have applied the idea of hyperdimensional computing to diverse learning and cognitive tasks, such as graph reasoning [22], robotics [23], neuromorphic computing [24], language recognition [25], genome sequencing [26] and activity recognition [9, 27]. Particularly, several recent works aimed to use HDC in area of biosignal sensors (i.e., EEG, EMG, or EXG sensors) for diverse tasks, including brain-computer interfaces, seizure identification, emotion detection, activity recognition, and gesture detection [9–13]. These methods often provide comparable accuracy to existing machine learning models while requiring a significantly lower computational cost. However, all existing HDC algorithms use a linear encoder with non-hardware-friendly operations. This makes HDC very inefficient as it requires to use of large dimensionality to solve realistic problems. To the best of our knowledge, TempHD is the first effort to design a highly accurate and efficient HDC classification for biosignal. Our approach maps data points in a non-linear manner with higher separability in high-dimensional space. This results in higher classification accuracy as well as the capability to learn from fewer samples.

Prior works also proposed different hardware acceleration for HDC, using ASIC [28–30] and processing in-memory [31–34]. However, these hardware designs are usually not commercially available and need a relatively long period to synthesize and fabricate after deriving the new applications. As such, to ease the deployment of the HDC in the real world, we need a framework solution to run HDC on a highly parallel but general-purpose platform, especially embedded CPU. However, the current HDC algorithms rely on costly high-dimensional operations, which are significantly slower than traditional processors. In contrast, we designed a novel HDC classification with hardware-friendly operations. Our solution leverages hardware-friendly mathematics for encoding with significant potential of acceleration in hardware.

7 CONCLUSION

In this paper, we propose TempHD, a novel hyperdimensional computing method for efficient and accurate biosignal classification. We first develop a novel non-linear hyperdimensional encoding that maps data points into high-dimensional space. Unlike existing HDC solutions that use costly mathematics for encoding, TempHD preserves spatial-temporal information of data in original space before mapping data into high-dimensional space. To obtain the most informative representation, our encoding method considers the non-linear interactions between both spatial sensors and temporally sampled data. Our evaluation shows that TempHD provides higher classification accuracy, significantly higher computation efficiency as well as the capability to learn from partially labeled data.

ACKNOWLEDGEMENTS

This study is supported by the National Science Foundation (NSF) #2127780, Semiconductor Research Corporation (SRC) Task No. 2988.001, Department of the Navy, Office of Naval Research, grant #N00014-21-1-2225, Air Force Office of Scientific Research, grant #22RT0060, and generous gift from Cisco.

REFERENCES

[1] R. R. Shrivastwa *et al.*, "A brain–computer interface framework based on compressive sensing and deep learning," *IEEE Consumer Electronics Magazine*, vol. 9, no. 3, pp. 90–96, 2020.

[2] Y. Su *et al.*, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," in *ACM SIGKDD KDD*, 2019.

[3] Y. Qin *et al.*, "A dual-stage attention-based recurrent neural network for time series prediction," *arXiv preprint arXiv:1704.02971*, 2017.

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

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

[6] A. Kazemi *et al.*, "Mimhd: Accurate and efficient hyperdimensional inference using multi-bit in-memory computing," in *ISLPED*. IEEE, 2021, pp. 1–6.

[7] Z. Zou *et al.*, "Scalable edge-based hyperdimensional learning system with brain-like neural adaptation," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2021, pp. 1–15.

[8] A. Hernández-Cano *et al.*, "Prid: Model inversion privacy attacks in hyperdimensional learning systems," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2021, pp. 553–558.

[9] A. Moin *et al.*, "A wearable biosensing system with in-sensor adaptive machine learning for hand gesture recognition," *Nature Electronics*, 2021.

[10] A. Rahimi *et al.*, "Hyperdimensional biosignal processing: A case study for emg-based hand gesture recognition," in *ICRC*. IEEE, 2016, pp. 1–8.

[11] U. Pale *et al.*, "Systematic assessment of hyperdimensional computing for epileptic seizure detection," *arXiv preprint arXiv:2105.00934*, 2021.

[12] E.-J. Chang *et al.*, "Hyperdimensional computing-based multimodality emotion recognition with physiological signals," in *AICAS*. IEEE, 2019, pp. 137–141.

[13] F. Asgarinejad *et al.*, "Detection of epileptic seizures from surface eeg using hyperdimensional computing," in *EMBC*. IEEE, 2020, pp. 536–540.

[14] "Monitoring error-related potentials." 2020. [Online]. Available: http://bnci-horizon-2020.eu/database/data-sets

[15] A. Rahimi, B. Recht *et al.*, "Random features for large-scale kernel machines." in *NIPS*, vol. 3, no. 4. Citeseer, 2007, p. 5.

[16] A. Rahimi and B. Recht, "Weighted sums of random kitchen sinks: replacing minimization with randomization in learning." in *Nips*. Citeseer, 2008, pp. 1313–1320.

[17] D. J. McFarland *et al.*, "Spatial filter selection for eeg-based communication," *Electroencephalography and clinical Neurophysiology*, vol. 103, no. 3, pp. 386–394, 1997.

[18] M. Seeck *et al.*, "The standardized eeg electrode array of the ifcn," *Clinical Neurophysiology*, vol. 128, no. 10, pp. 2070–2077, 2017.

[19] P. W. Ferrez *et al.*, "You are wrong!—automatic detection of interaction errors from brain waves," in *IJCAI*, no. CONF, 2005.

[20] P. Ferrez *et al.*, "Error-related eeg potentials generated during simulated brain computer interaction," *IEEE T-BME*, vol. 55, no. 3, 2008.

[21] A. Rahimi *et al.*, "Hyperdimensional computing for blind and one-shot classification of eeg error-related potentials," *Mobile Networks and Applications*, 2020.

[22] P. Poduval *et al.*, "Graphd: Graph-based hyperdimensional memorization for brain-like cognitive learning," *Frontiers in Neuroscience*, p. 5, 2022.

[23] A. Mitrokhin *et al.*, "Learning sensorimotor control with neuromorphic sensors: Toward hyperdimensional active perception," *Science Robotics*, vol. 4, no. 30, 2019.

[24] Z. Zou, *et al.*, "Eventhd: Robust and efficient hyperdimensional learning with neuromorphic sensor," *Frontiers in Neuroscience*, vol. 16, 2022.

[25] A. Rahimi *et al.*, "A robust and energy-efficient classifier using brain-inspired hyperdimensional computing," in *ISLPED*, 2016, pp. 64–69.

[26] Z. Zou *et al.*, "Biohd: an efficient genome sequence search platform using hyperdimensional memorization," in *Proceedings of the 49th Annual International Symposium on Computer Architecture*, 2022, pp. 656–669.

[27] Y. Kim *et al.*, "Efficient human activity recognition using hyperdimensional computing," in *IOT*, 2018, pp. 1–6.

[28] S. Datta *et al.*, "A programmable hyper-dimensional processor architecture for human-centric iot," *JETCAS*, vol. 9, no. 3, pp. 439–452, 2019.

[29] B. Khaleghi *et al.*, "Shear er: highly-efficient hyperdimensional computing by software-hardware enabled multifold approximation," in *ISLPED*, 2020, pp. 241–246.

[30] Y. Ni *et al.*, "Algorithm-hardware co-design for efficient brain-inspired hyperdimensional learning on edge," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2022, pp. 292–297.

[31] G. Karunaratne *et al.*, "In-memory hyperdimensional computing," *Nature Electronics*, vol. 3, no. 6, pp. 327–337, 2020.

[32] M. Imani *et al.*, "Searchd: A memory-centric hyperdimensional computing with stochastic training," *TCAD*, vol. 39, no. 10, pp. 2422–2433, 2019.

[33] T. F. Wu *et al.*, "Brain-inspired computing exploiting carbon nanotube fets and resistive ram: Hyperdimensional computing case study," in *ISSCC*. IEEE, 2018.

[34] M. Imani *et al.*, "Exploring hyperdimensional associative memory," in *HPCA*. IEEE, 2017, pp. 445–456.