Unleashing Hyperdimensional Computing with Nyström Method based Encoding

Quanling Zhao, Anthony Thomas, Ari Brin, Xiaofan Yu, Tajana Rosing
Department of Computer Science and Engineering
University of California San Diego
{quzhao,ahthomas,abrin,x1yu,tajana}@ucsd.edu

Abstract

Hyperdimensional computing (HDC) is an approach for solving cognitive information processing tasks using data represented as high-dimensional, low-precision, vectors. The technique has a rigorous mathematical backing, and is easy to implement in energy-efficient and highly parallelizable hardware like FPGAs and "in-memory" architectures. The success of HDC based machine learning approaches is heavily dependent on the mapping from raw data to high-dimensional space. In this work, we propose a new method for constructing this mapping that is based on the Nyström method from the literature on kernel approximation. Our approach provides a simple recipe to turn any user-defined positive-semidefinite similarity function into an equivalent mapping in HDC. There is a vast literature on the design of such functions for learning problems, and our approach provides a mechanism to import them into the HDC setting, potentially expanding the types of problems that can be tackled using HDC. An empirical comparison of our approach against existing HDC encoding methods on a variety of classification tasks shows that we can achieve 10%-37% and 3%-18% better classification accuracy on graph and string datasets respectively.

1 Introduction

Biological brains "compute" using representations of data that are intrinsically fault tolerant, amenable to highly-parallel circuitry, and expose complex structure in the environment in a way that is easy to learn from [10]. Motivated by these desirable qualities, hyperdimensional computing (HDC) builds on theories of representation from cognitive science [16, 33] in an effort to develop a new approach to designing hardware and algorithms for information processing tasks [16]. In HDC, all computation is performed using high-dimensional, low-precision, vector representations of data. These representations can be manipulated using simple, element-wise operators, so as to implement learning or other information processing tasks.

In contrast to deep learning models, training HDC based models can typically be done in a single pass over the training data [49] and typically do not require back-propagation. The operations used in HDC are lightweight and highly parallelizable, making them suitable for implementation on low-energy and highly-parallel hardware platforms, which makes it an attractive alternative for implementing learning in resource constrained settings. As a result, HDC has gained significant interest in recent years, especially in Internet of Things (IoT) [19, 51, 29]; and in the computer hardware community [6, 17, 20] such as FPGAs [38], GPUs [18], ASICs [50] and in-memory computing [6, 48].

The first stage in any HDC task is encoding, which maps data from its ambient representation $x \in \mathcal{X}$, into a representation $\phi(x)$ that lives in some high-dimensional inner-product space $\mathcal{H}$ [44]. Architectures for HDC supply a pair of operators for addition and multiplication (called 'bundling"

37th First Workshop on Machine Learning with New Compute Paradigms at NeurIPS 2023(MLNPCP 2023).

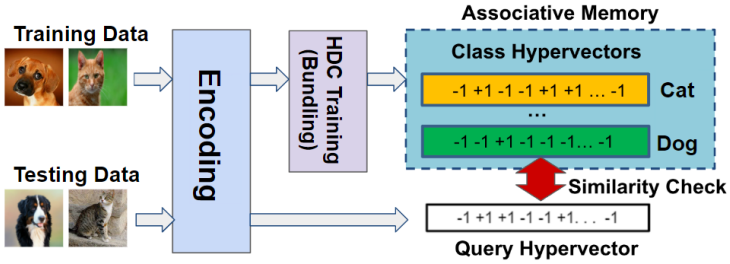

Figure 1: Overview of HDC training and inference for classification tasks.

and "binding" in the HDC literature), which can be used to build representations of complex structures from simple building blocks or to implement tasks like learning. For instance, a popular approach classification using HDC is to represent each class as a bundle (sum) of the encodings of its training data, called a "prototype," whereupon inference can be performed by finding the closest prototype to a query.

The crucial assumption underlying the success of this technique is that "similar" points in $\mathcal{X}$ are mapped to "similar" regions of $\mathcal{H}$. In practice, this desideratum typically means that dot-products in $\mathcal{H}$ should be reflective of some salient notion of similarity on $\mathcal{X}$. In HDC, one typically builds representations incrementally, by bundling and binding together the embeddings of simpler atoms. For instance, to embed a feature vector, one might bundle together embeddings of the individual features. In this work, we observe that one can also go in the opposite direction: starting from a known similarity function of interest, it is possible to generate an equivalent—up to some approximation factor—encoding function.

The advantage of this "top down" approach is that there is a vast literature on designing good similarity functions for different kinds of learning problems. In machine learning, similarity functions that work by computing inner-products between high-dimensional embeddings of data are called kernel functions, and are the basis of kernel methods, a vast area of research in theoretical and applied ML [40, 42]. This literature has devoted substantial attention to the problem of designing good similarity functions, which are also potentially applicable to the kinds of problems encountered in HDC. In this work, we study an approach, based on the Nyström method from the literature on kernel approximation [46], which can take a user defined kernel and generate an low-precision, randomized embedding, suitable for use in the kinds of learning algorithms employed in HDC. In a nutshell, the contributions of this paper are as follows:

- We propose a new way to generate embeddings for HDC which can turn any user-defined positive-semidefinite similarity function into an equivalent embedding.
- We analyze the similarity-preserving properties of our encoding method formally.
- We evaluate our encoding method empirically across a variety of tasks, including string, graph, and image classification. The results show that our method achieves 10%-37% and 3%-18% better classification accuracy on graph and string classification, respectively.

From a practical standpoint, our work has the potential to increase performance on HDC tasks like classification and regression by allowing practitioners to access the large repertoire of kernels that have been designed for these tasks.

2 Background

2.1 Learning with HDC

In this work, we focus on using HDC to solve classification problems, which is a common practical application of the technique [12, 35, 26]. Fig 1 demonstrates a typical HDC learning workflow for classification [28, 31, 27]. Let $\mathcal{D} = \{(x_1, y_1), ..., (x_n, y_n)\}$ be a set of training data, where $x_i \in \mathcal{X}$ is an input, and $y_i \in \{1, ..., c\}$ is a class label. The first step is to embed the training data into a d-dimensional inner-product space $\mathcal{H}$ under a map (called the encoding function) $\phi : \mathcal{X} \to \mathcal{H}$. The "training" step then associates each class with a vector in $\mathcal{H}$, which is typically formed by summing

(bundling) the training data corresponding to a particular class. That is, one represents class j as:

$$\theta_j = \sum_{i=1}^{n} \alpha_{ij} \phi(x_i),$$

where $\alpha_i = \mathbb{1}(y_i = j)$. In practice, one sometimes quantizes θ_j in some fashion to reduce precision, which is beneficial in some hardware settings. It is also common to fine tune the θ_j's by running the Perceptron algorithm [37]. The label of a query x is predicted by via:

$$\hat{y} = \underset{j \in \{1,...,c\}}{\arg\max} \langle \phi(x), \theta_j \rangle,$$

where the operands are sometimes normalized in some fashion if appropriate. In any event, this procedure can be interpreted as associating each class to a linear function in $\mathcal{H}$.

A number of HDC encoding methods have been proposed to encode various types of data. For example, string or text data can be encoded into hypervectors through the N-gram encoding method [15, 13]. Concretely, let $S = (a_1 a_2 a_3 ... a_N)$ be a string of N characters drawn from some alphabet $\mathcal{A}$ (say, the English-language alphabet or $\{\texttt{A,T,G,C}\}$). To encode S, we start by assigning each $a \in \mathcal{A}$ an embedding $\phi(a)$ by sampling uniformly at random from $\mathcal{H}$, after which, one might represent S as $a_1 \circ \rho(a_2) \circ \rho^2(a_3) \circ \cdots \circ \rho^{N-1}(a_N)$ where $\circ$ can be either element-wise addition [13] or multiplication [15] and ρ is a permutation operation of the vector coordinates, ρ^n means same permutation applied n times in sequence. If $\circ$ is multiplication, the way N-grams representations are constructed makes them almost mutually orthogonal in $\mathcal{H}$ in the sense that $\mathbb{E}[\phi(S) \cdot \phi(S')] = \mathbb{1}(S = S')$. The deviation from this expectation can be controlled using concentration arguments [44].

Finally, longer strings that contain multiple N-grams can be treated as a bundle of N-gram vectors. This string encoding scheme can be viewed as a "compressed" version of the bag-of-words model [8] in the sense that all N-gram vectors are superimposed into one representation. Intuitively, the inner-product between two such encoded strings can measure how "similar" two strings are, as that value would be large if two strings shared many common N-grams, and vice versa. For general feature vectors (or simple images), techniques based on random projection are popular [28, 19]. These methods tend to capture fairly simple notions of similarity based on the L1/L2 or angular distance. However, the design of good encoding functions for complex forms of data like graphs and time-series remains an important area of research. For inspiration, we turn to another area of machine learning that has thought extensively about how to measure similarities between data points using high-dimensional vectors.

2.2 Kernel Methods

Kernel methods are a wide ranging area of research in statistics and machine learning that shares many similarities with HDC [40, 25, 11]. Just like in HDC, kernel methods work by embedding data into a high-dimensional space wherein similarities are measured using inner-products. That is to say, kernel methods measure similarities between data points $x, x' \in \mathcal{X}$ via a function $K(x, x') = \langle \psi(x), \psi(x') \rangle$, called a "kernel function," where $\psi : \mathcal{X} \to \mathcal{H}$ is an embedding into an inner-product space. For many types of kernel functions used in practice, it is possible to compute $K(x, x')$ directly on the ambient representation of the data without materializing the embeddings. Notable examples include the Gaussian kernel $K(x, x') \propto \exp(-\|x - x'\|_2^2)$, and the p-th order polynomial kernel $K(x, x') = (1 + x^T x')^p$. Both of these kernels can be evaluated in closed form on the ambient representation of the data, allowing kernel methods to *implicitly* compute a similarity based on a high-dimensional embedding. HDC, however, always explicitly materializes the embeddings, hence the need for an encoding function ϕ.

Kernel based learning methods make predictions using functions taking the form $f(x) = \sum_{i=1}^{n} a_i k(x_i, x)$, where $x_1, ..., x_n$ are training data points, and $\alpha_1, ..., \alpha_n$ are weights that are learned by a training algorithm. Noting that $f(x) = \sum_{i=1}^{n} \alpha_i k(x, x_i) = \psi(x) \cdot \theta$ where $\theta = \sum_{i=1}^{n} a_i \psi(x_i)$, in this way we can interpret such functions as *linear models* in the embedding space associated with the kernel, much like in the previous paragraph on HDC. One significant difference between kernel methods and HDC, is that in the former the embeddings are implicit, and similarities are evaluated using the kernel function. This property is appealing because it allows one to efficiently work with infinite-dimensional embeddings, which can have desirable properties for learning [43].

2.3 Kernel methods and HDC

There is a large theoretical and applied literature on kernel methods that has designed kernels (e.g. similarity functions) that are applicable to many settings of practical interest [30, 24, 41, 40]. To give a concrete example of how the literature on kernel methods can offer insights for the HDC community, we consider the method for encoding graphs presented in [31]. GraphHD [31] proposes to first use the PageRank centrality metric [4] to rank the "importance" of each node and assign a hypervector to each node based on its importance, in the sense that nodes in different graphs with the same ranking will be assigned the same hypervector. Edges are represented by multiplying (binding) of two node hypervectors, and an entire graph can be represented as the summation (bundle) of edge hypervectors. This encoding method is limited in the sense that only topological information of graphs is preserved during encoding, while other crucial information such as node labels or node attributes (where each node in the graph is associated with either a feature vector or a label) is not being utilized. Such limitations, however, have been addressed with kernel methods. For example, the propagation kernel [30] is able to work with graphs that have node labels or node attributes. The propagation kernel uses random walk techniques to measure not only the structural difference between graphs but also taking node labels or node attributes into consideration as well, therefore is capable of capturing a potentially richer notion of similarity.

On time-series data, the permutation operation is used in a similar fashion to encode the temporal order [2]. However, such encoding method can fail if the events between two time-series do not align exactly. Since each time step is associated with a unique permutation during encoding, if events in time-series are shifted, even by a small amount, existing HDC encoding methods will map two time-series to nearly orthogonal vectors. In practice, however, if two time-series data reflect the same underlying activity or nature, one would want that similarity to be preserved after the encoding, even if some event mis-alignment may exist. Here again, the literature on kernel methods suggests a solution: the dynamic-time-warping kernel [7] can handle time-series sequences with dis-alignment or with time-stretching/compression. In this work, our goal is to devise a procedure that can translate any kernel into an equivalent HDC encoding, thereby allowing practitioners to exploit the wealth of kernel functions that have been designed for practical problems while continuing to reap the benefits of computing with distributed representations.

2.4 Related Work

The connection between HDC and kernel approximation is generally well known [44, 32, 49], mostly through the lens of random Fourier features (RFF) [36]. RFF is a sampling based scheme that generates a vector of features $\phi(x) \in \mathbb{R}^d$ with the property that $\phi(x) \cdot \phi(x') \approx K(x, x')$, where K is a shift-invariant kernel (e.g. the Gaussian kernel, polynomial kernel). Closely related methods arise in the HDC literature under the names "nonlinear-encoding" [27, 14] and "fractional power encoding" [32]. Using RFF in the context of HDC means that inner-products in HD space approximate some shift-invariant kernel, usually the Gaussian kernel. However, a limitation of RFF is that it can only work with shift-invariant kernels on a Euclidean space, which many useful kernels do not satisfy (i.e. kernels on graphs and strings). Our method provides a way to generate approximations for a larger class of kernels that do not need to be translation invariant.

3 HDC Encoding using the Nyström Approximation

In this section we describe our new encoding algorithm for HDC using Nyström method for kernel approximation. We also show formally that the inner-product between encoded samples preserves normalized kernel values.

Formally, given the dataset $\mathcal{D}$ and a suitable kernel function K, the goal is to generate an encoding function $\phi : \mathcal{X} \to \mathcal{H}$ such that $\langle \phi(x), \phi(x') \rangle \approx K(x, x')$, while ensuring that the representation in $\mathcal{H}$ remains high-dimensional, which is an important property making HDC representation noise-robust and hardware efficient [44, 21]. Having embeddings that approximate a particular kernel is important as it enables HDC learning algorithms to exploit more useful similarities induced by the kernel.

3.1 Nyström method

Fitting kernel machines commonly requires storing all pairwise evaluations of the kernel function in a large matrix K defined element-wise by $K_{ij} = K(x_i, x_j)$, problematic when n is large. The Nyström method is a low-rank matrix approximation technique widely employed to speed up kernel machines by avoiding the need to evaluate the entire kernel matrix [46, 5, 22]. In a way, Nyström method is similar to RFF since they are both sampling based schemes and can be used to approximate kernel functions. The key difference between RFF and Nyström method is that Nyström method can work with a larger class of kernels than RFF, many of which are useful for applications involving discrete structures such as strings, graphs and more.

Intuitively, the Nyström method works by sub-sampling the kernel matrix and reconstructing the full kernel matrix from the sampled one. This is possible because the kernel matrix is typically close to low-rank in practice. Concretely, suppose we have a dataset $\mathcal{D} = \{x_1, x_2, ..., x_n\}$, from which we sample a set of landmarks $\mathcal{Z} = \{z_1, ..., z_s\}$ uniformly at random, where $s \ll n$. Let $G \in \mathbb{R}^{n \times n}$ be the full kernel matrix defined element-wise by $G_{ij} = K(x_i, x_j)$, and let $H_{\mathcal{Z}} \in \mathbb{R}^{s \times s}$ be the sub-sampled kernel matrix defined element-wise by $(H_{\mathcal{Z}})_{ij} = K(z_i, z_j)$. The Nyström method yields the approximation [5]:

$$\hat{G} = C H_{\mathcal{Z}}^{+} C^T \approx G,$$

Where C is an $n \times s$ matrix such that $C_{ij} = K(x_i, z_j)$ for some kernel function K, and H_Z^+ denotes the pseudo-inverse of $H_{\mathcal{Z}}$. That is, let Q and Λ be the eigenvectors and eigenvalues of $H_{\mathcal{Z}}$ then:

$$H_Z^+ = Q \Lambda^{-1} Q^T \quad \text{(symmetric eigen-decomposition)}$$

To further decompose Eq. (3.1), let $C^{(i)}$ denotes i^{th} row of C in a column vector, we can have embedding for each data point such that:

$$\hat{G_{ij}} \approx \phi_{nys}(x_i)^T \phi_{nys}(x_j) = \left(\Lambda^{-\frac{1}{2}} Q^T C^{(i)}\right)^T \left(\Lambda^{-\frac{1}{2}} Q^T C^{(j)}\right) \tag{1}$$

In general, the embeddings generated by Nyström method do not need to be high-dimensional to have the similarity preserving property, however, they may lack other desiderata of HDC like noise robustness, and low-precision. Our method rectifies this issue by composing the features extracted using the Nyström method with another encoding technique that preserves angular similarities which we discuss in detail in section 3.2.

Due to the data-dependent nature of the Nyström method, the quality of its approximation and computational complexity is highly dependent on the quality and size of $\mathcal{Z}$. As an initial step, in this paper we simply use uniform sampling without replacement as our sampling strategy. However, more sophisticated strategies such as ensemble and adaptive sampling [22] for constructing landmark sets can potentially yield better performance, and it would be of interest to explore these in future work.

3.2 Encoding Process

To achieve the aforementioned goals (generate HDC embeddings such that their inner-products approximate some useful kernel) we compose random projection with Nyström method. Alg. 1 details the generating process of Nyström random projection. After which, the encoding of a data point can be done through Alg. 2.

Algorithm 1 Generate Nyström Random Projection

Require: kernel K over $\mathcal{X}$, dataset $\mathcal{D}$, number of landmarks $s > 0$, HDC dimension $d > 0$
$\mathcal{Z} \leftarrow$ uniform sampling of s data points from $\mathcal{D}$ ***/*Landmarks*/***
$(H_{\mathcal{Z}})_{ij} = K(z_i, z_j) \quad \forall \quad 0 \leq i, j \leq s$ ***/*Partial kernel Matrix over landmarks*/***
$Q \Lambda Q^T = H_{\mathcal{Z}}$ ***/*Symmetric Eigen-decomposition*/***
$P_{rp} = [w_1, w_1, \cdots, w_d]^T \in \mathbb{R}^{d \times s}$ ***/*w_i sampled from s dimensional unit sphere*/***
$P_{nys} = P_{rp} \Lambda^{-\frac{1}{2}} Q^T$
return P_{nys} , $\mathcal{Z}$

Algorithm 2 Encode one data point from $\mathcal{X}$

Require: $x_i \in \mathcal{X}$, Projection P_{nys}, Landmarks $\mathcal{Z}$ and kernel function K
$C^{(i)} = [K(x_i, z_1) \quad K(x_i, z_2) \quad \cdots \quad K(x_i, z_s)]^T \in \mathbb{R}^s$
return $\sqrt{\frac{\pi}{2d}}$ sign $\left(P_{nys} C^{(i)}\right)$

The rows of $P_{rp} \in \mathbb{R}^{d \times s}$ are sampled from the uniform distribution over the s dimensional unit sphere. This allows us to use the following result regarding sign-thresholded random projection - suppose $v, v' \in \mathbb{R}^n$ are unit vectors, with respect to randomness in the sampling of P_{rp}, the following expectation holds:

$$\mathbb{E}\left[\frac{1}{d}\text{sign}(Pv) \cdot \text{sign}(Pv')\right] = (1 - 2\cos^{-1}(v \cdot v')/\pi) \quad \text{(Extension of Corollary 19 in [44])} \tag{2}$$

Our proposed encoding method generates hypervectors for which it is true that the similarity induced by the kernel K is preserved by dot product between encodings in $\mathcal{H}$. We summarize this result in the following informal theorem:

Theorem 1 *Define $\Theta_i = \Lambda^{-\frac{1}{2}} Q^T C^{(i)}$ and $\Theta_i \in \mathbb{R}^n$. Based on above HDC encoding algorithm, the encoding function $\phi : \mathcal{X} \to \mathcal{H}$ can be write as following:*

$$\phi(x_i) = \sqrt{\frac{\pi}{2d}} \textit{sign}(P_{rp}\Theta_i) \tag{3}$$

We pose no restriction on Kernel K, the following holds up to a first order approximation:

$$\mathbb{E}\left[\langle \phi(x_i), \phi(x_j) \rangle\right] \approx \frac{\hat{G}_{ij}}{\sqrt{\hat{G}_{ii}\hat{G}_{jj}}} \quad \textit{i.e. normalized kernel} \tag{4}$$

Where $\hat{G}_{ij}$ is estimated kernel value between x_i and x_j produced by Nyström method and the expectation is taken with respect to randomness and orthogonality in P_{rp}.

Proof: Let $\hat{\Theta}_i \in \mathbb{R}^s$ denote $\frac{\Theta_i}{||\Theta_i||}$. Noting that $\text{sign}(cv) = \text{sign}(v)$ for any $v \in \mathbb{R}^s$ and $c > 0$:

$$\begin{aligned}
\langle \phi(x_i), \phi(x_j) \rangle &= \left(\sqrt{\frac{\pi}{2}}\sqrt{\frac{1}{d}}\text{sign}\left(P_{rp}\Theta_i\right)^T \sqrt{\frac{\pi}{2}}\sqrt{\frac{1}{d}}\text{sign}\left(P_{rp}\Theta_j\right)\right) \\
&= \frac{\pi}{2}\left(\frac{1}{d}\text{sign}\left(P_{rp}\frac{\Theta_i}{||\Theta_i||}\right)^T \text{sign}\left(P_{rp}\frac{\Theta_j}{||\Theta_j||}\right)\right) \\
&= \frac{\pi}{2}\left(\frac{1}{d}\text{sign}\left(P_{rp}\hat{\Theta}_i\right)^T \text{sign}\left(P_{rp}\hat{\Theta}_j\right)\right)
\end{aligned} \tag{5}$$

Using result regarding sign-thresholded random projection in equation. 2, we have:

$$\mathbb{E}\left[\langle \phi(x_i), \phi(x_j) \rangle\right] \approx \frac{\pi}{2}\left(1 - \frac{2\cos^{-1}\left(\hat{\Theta}_i^T \hat{\Theta}_j\right)}{\pi}\right) \tag{6}$$

Since $||\hat{\Theta}_i|| = ||\hat{\Theta}_j|| = 1$, it follows $-1 \leq \hat{\Theta}_i^T \hat{\Theta}_j \leq 1$, which is within the domain of $\cos^{-1}$. Use first order Taylor series of $\cos^{-1}(x)$ that $\cos^{-1}(x) \approx \frac{\pi}{2} - x$:

$$\mathbb{E}\left[\langle \phi(x_i), \phi(x_j) \rangle\right] \approx \frac{\pi}{2}\left(1 - \frac{2\left(\frac{\pi}{2} - \hat{\Theta}_i^T \hat{\Theta}_j\right)}{\pi}\right) \tag{7}$$

Recall Nyström method in equation 1: ${\Theta_i}^T\Theta_j = (\Lambda^{-\frac{1}{2}}Q^TC^{(i)})^T(\Lambda^{-\frac{1}{2}}Q^TC^{(j)}) = \hat{G}_{ij}$ and $||\Theta_i|| = \sqrt{{\Theta_i}^T\Theta_i} = \sqrt{\hat{G}_{ii}}$:

$$\hat{\Theta}_i^T\hat{\Theta}_j = \frac{{\Theta_i}^T\Theta_j}{||\Theta_i|| \cdot ||\Theta_j||} = \frac{\hat{G}_{ij}}{\sqrt{\hat{G}_{ii}\hat{G}_{jj}}} \tag{8}$$

Finally:

$$\mathbb{E}\left[\langle\phi(x_i), \phi(x_j)\rangle\right] \approx \frac{\pi}{2}\left(1 - \frac{2\left(\frac{\pi}{2} - \frac{\hat{G}_{ij}}{\sqrt{\hat{G}_{ii}\hat{G}_{jj}}}\right)}{\pi}\right) = \frac{\hat{G}_{ij}}{\sqrt{\hat{G}_{ii}\hat{G}_{jj}}} \tag{9}$$

Thus, our proposed encoding method preserves the kernel in $\mathcal{H}$ in the sense that the inner-product between any pair of encoded data points approximates the normalized kernel value of some user defined kernel K. Since HDC uses inner-product based metric in $\mathcal{H}$ for inference as explained in section 2.1, the similarity metric kernel K induces is also preserved and will be beneficial for any subsequent HDC learning algorithms.

4 Experiments

We conduct empirical experiments with our proposed HDC encoding method in comparison with existing HDC encoding methods. To ensure the fairness of comparison, we use the identical HDC learning pipeline adapted from [9] when evaluating different encoding methods.

4.1 Experimental setup

Table 1: Summary of tasks and datasets

Task	dataset	# of training samples	# of testing samples	# of classes
String	Protein sequences [39]	721	181	6
	SMS Spam collection [1]	4459	1115	2
Graph	ENZYME [1] [3]	480	120	6
	NCI1 [2] [45]	3288	822	2
Image	MNIST [23]	60000	10000	10
	FashionMNIST [47]	60000	10000	10

Baselines: The encoding method we propose is general-purpose and can be applied to a wide range of data and tasks as long as a suitable kernel function exist. To confirm its versatility, we conduct assessments across three distinct tasks: string, graph and image classification. In each task, we test on two representative datasets, and a summary of each dataset is shown in Table 1. We benchmark our method against the prevalent HDC encoding baselines within that domain. Here, we summarize the tasks and baseline HDC encoding methods we plan to compare:

- **String classification** using N-gram HDC encoding approach. The details of N-gram HDC encoding are detailed in section 2.1, here we follow the work of Imani et al [13] where element-wise addition (bundling) is used for generating N-gram hypervectors.
- **Graph classification** using graph encoding scheme introduced in GraphHD [31]. The details of this encoding procedure are discussed in section 2.2.
- **Image classification** using linear [34, 12] and nonlinear [44] random projection.

For our method, we use the spectrum kernel [24] for string classification, which computes the number of unique N-grams between two input sequences explicitly. Propagation kernel [30] is used for graph

[1] Attributed graphs: each node is associated with a feature vector.

[2] Labelled graphs: each node is associated with a label to indicate the node type.

classification for its ability to work with labelled or attributed graphs as discussed in section 2.2. For image classification, Gaussian kernel is used. We choose the aforementioned kernels for their relatively low computation complexity and effectiveness on respective tasks. We set the number of landmarks as $s = max(300, 2\%$ of training data) on each dataset.

4.2 Results

Table 2: Experimental results and comparison with other HDC encoding methods. GraphHD results obtained from original paper.[31]. Best accuracy result for each dataset are marked with underscore.

Task	**Dataset**	**Ours**	**N-gram** [13]	**GraphHD** [31]	**LinearRP** [34, 12]	**NonlinearRP** [44]
String	Protein sequence	**99%**	81%	-	-	-
	SMS Spam collection	**96%**	93%	-	-	-
Graph	ENZYME	**63%**	-	26%	-	-
	NCI1	**72%**	-	62%	-	-
Image	MNIST	96%	-	-	96%	**97%**
	FashionMNIST	**86%**	-	-	85%	**86%**

The accuracy results on three tasks are summarized in Table 2. In comparison with the existing HDC encoding techniques, our proposed encoding method is able to achieve better accuracy results on string and graph classification tasks. Particularly, our method achieves 10%-37% and 3%-18% better classification accuracy on graph and string classification, respectively. The improvements are especially significant on the ENZYME dataset, as previous HDC encoding method on graph can not utilize node attributes in fully attributed graphs but can be done with our method. On string datasets, our method again consistently achieves better accuracy when compared with N-gram based HDC encoding. The choice of data-appropriate kernel is important: on graph datasets, we use propagation kernel [30] which quantifies the structural similarity between graphs through information propagation, thus providing a more effective similarity metric to GraphHD [31]. On the other hand, the performance on MNIST and FashionMNIST are nearly identical between our method and existing methods, with nonlinear encoding (RFF) performing slightly better. In this case we are not able to surpass the existing linear and nonlinear random projection encoding because the structure of MNIST and FashionMNIST can already be well preserved through a linear kernel or Gaussian kernel. Overall, the strength of our HDC encoding method becomes most apparent when dealing with data with complex structure or data with attributes that traditional HDC encoding methods do not exploit, for example, node attributes and labels in graph.

5 Discussion

This work allows future HDC works to exploit the power of kernel methods while still conforming to the general formalism and benefits of HDC. However, even though our method is showing promising results on small scale datasets, its applicability and effectiveness on larger scale tasks, which is the area where HDC excels, are yet to be determined. We recognize that the improvements in our proposed HDC encoding methods also come with additional computation costs in the form of kernel evaluation. How to minimize such cost and how to construct landmark set in a more principled way to achieve the best efficiency-accuracy trade-offs for HDC applications are non-trivial problems that need further investigations.

6 Summary

In this paper, we leverage the connection between the kernel method and HDC through the lens of Nyström method for kernel estimation. In particular, we propose a new encoding method for HDC that can construct encoding functions using suitable kernel functions for specific tasks. We further provide a formal proof showing the embeddings generated through our method indeed approximate normalized kernel values, therefore enabling HDC to capture a rich notion of similarity associated with the kernel. In our experiments, we evaluate our encoding method empirically across a variety of tasks. Compared with existing HDC encoding methods, results show that our method achieves

10%-37% and 3%-18% better classification accuracy on graph and string classification, respectively. Considering the wide variety of existing kernel functions, our method may pave the way to exploring the parallels between HDC and kernel methods for future HDC works.

7 Acknowledgement

This work was supported in part by National Science Foundation under Grants #2003279, #1826967, #2100237, #2112167, #1911095, #2112665, and in part by SRC under task #3021.001. This work was also supported in part by PRISM and CoCoSys, centers in JUMP 2.0, an SRC program sponsored by DARPA.

References

[1] Tiago Almeida and Jos Hidalgo. SMS Spam Collection. UCI Machine Learning Repository, 2012. DOI: https://doi.org/10.24432/C5CC84.

[2] Fatemeh Asgarinejad, Anthony Thomas, and Tajana Rosing. Detection of epileptic seizures from surface eeg using hyperdimensional computing. In *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pages 536–540. IEEE, 2020.

[3] Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1):i47–i56, 2005.

[4] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998.

[5] Petros Drineas, Michael W Mahoney, and Nello Cristianini. On the nyström method for approximating a gram matrix for improved kernel-based learning. *journal of machine learning research*, 6(12), 2005.

[6] Arpan Dutta, Saransh Gupta, Behnam Khaleghi, Rishikanth Chandrasekaran, Weihong Xu, and Tajana Rosing. Hdnn-pim: Efficient in memory design of hyperdimensional computing with feature extraction. In *Proceedings of the Great Lakes Symposium on VLSI 2022*, pages 281–286, 2022.

[7] Steinn Gudmundsson, Thomas Philip Runarsson, and Sven Sigurdsson. Support vector machines and dynamic time warping for time series. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 2772–2776. IEEE, 2008.

[8] Zellig S Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

[9] Alejandro Hernández-Cano, Namiko Matsumoto, Eric Ping, and Mohsen Imani. Onlinehd: Robust, efficient, and single-pass online learning using hyperdimensional system. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 56–61. IEEE, 2021.

[10] John A Hertz. *Introduction to the theory of neural computation*. Crc Press, 2018.

[11] Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. 2008.

[12] Mohsen Imani, Justin Morris, John Messerly, Helen Shu, Yaobang Deng, and Tajana Rosing. Bric: Locality-based encoding for energy-efficient brain-inspired hyperdimensional computing. In *Proceedings of the 56th Annual Design Automation Conference 2019*, pages 1–6, 2019.

[13] 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.

[14] Mohsen Imani, Saikishan Pampana, Saransh Gupta, Minxuan Zhou, Yeseong Kim, and Tajana Rosing. Dual: Acceleration of clustering algorithms using digital-based processing in-memory. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 356–371. IEEE, 2020.

[15] Aditya Joshi, Johan T Halseth, and Pentti Kanerva. Language geometry using random indexing. In *Quantum Interaction: 10th International Conference, QI 2016, San Francisco, CA, USA, July 20-22, 2016, Revised Selected Papers 10*, pages 265–274. Springer, 2017.

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

[17] Jaeyoung Kang, Behnam Khaleghi, Yeseong Kim, and Tajana Rosing. Xcelhd: An efficient gpu-powered hyperdimensional computing with parallelized training. In *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 220–225. IEEE, 2022.

[18] Jaeyoung Kang, Behnam Khaleghi, Tajana Rosing, and Yeseong Kim. Openhd: A gpu-powered framework for hyperdimensional computing. *IEEE Transactions on Computers*, 71(11):2753–2765, 2022.

[19] Behnam Khaleghi, Jaeyoung Kang, Hanyang Xu, Justin Morris, and Tajana Rosing. Generic: highly efficient learning engine on edge using hyperdimensional computing. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pages 1117–1122, 2022.

[20] Behnam Khaleghi, Sahand Salamat, Anthony Thomas, Fatemeh Asgarinejad, Yeseong Kim, and Tajana Rosing. Shear er: highly-efficient hyperdimensional computing by software-hardware enabled multifold approximation. In *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*, pages 241–246, 2020.

[21] Denis Kleyko, Dmitri Rachkovskij, Evgeny Osipov, and Abbas Rahimi. A survey on hyperdimensional computing aka vector symbolic architectures, part ii: Applications, cognitive models, and challenges. *ACM Computing Surveys*, 55(9):1–52, 2023.

[22] Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. Sampling methods for the nyström method. *The Journal of Machine Learning Research*, 13(1):981–1006, 2012.

[23] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[24] Christina Leslie, Eleazar Eskin, and William Stafford Noble. The spectrum kernel: A string kernel for svm protein classification. In *Biocomputing 2002*, pages 564–575. World Scientific, 2001.

[25] Giacomo Meanti, Luigi Carratino, Lorenzo Rosasco, and Alessandro Rudi. Kernel methods through the roof: handling billions of points efficiently. *Advances in Neural Information Processing Systems*, 33:14410–14422, 2020.

[26] Alisha Menon, Daniel Sun, Sarina Sabouri, Kyoungtae Lee, Melvin Aristio, Harrison Liew, and Jan M Rabaey. A highly energy-efficient hyperdimensional computing processor for biosignal classification. *IEEE Transactions on Biomedical Circuits and Systems*, 16(4):524–534, 2022.

[27] Victor Miranda and Olivia d'Aliberti. Hyperdimensional computing encoding schemes for improved image classification. In *2022 IEEE International Symposium on Technologies for Homeland Security (HST)*, pages 1–9. IEEE, 2022.

[28] J Morris et al. Hydrea: Towards more robust and efficient machine learning systems with hyperdimensional computing-based classifiers. *DATE'21*.

[29] Justin Morris, Kazim Ergun, Behnam Khaleghi, Mohsen Imani, Baris Aksanli, and Tajana Rosing. Hydrea: Towards more robust and efficient machine learning systems with hyperdimensional computing. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 723–728. IEEE, 2021.

[30] Marion Neumann, Roman Garnett, Christian Bauckhage, and Kristian Kersting. Propagation kernels: efficient graph kernels from propagated information. *Machine learning*, 102:209–245, 2016.

[31] Igor Nunes, Mike Heddes, Tony Givargis, Alexandru Nicolau, and Alex Veidenbaum. Graphhd: Efficient graph classification using hyperdimensional computing. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1485–1490. IEEE, 2022.

[32] E Paxon Frady, Denis Kleyko, Christopher J Kymn, Bruno A Olshausen, and Friedrich T Sommer. Computing on functions using randomized vector representations. *arXiv e-prints*, pages arXiv–2109, 2021.

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

[34] DA Rachkovskij. Formation of similarity-reflecting binary vectors with random binary projections. *Cybernetics and Systems Analysis*, 51:313–323, 2015.

[35] 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.

[36] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

[37] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[38] 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.

[39] Muthu Krishnan Selvaraj, Anamika Thakur, Manoj Kumar, Anil Kumar Pinnaka, Chander Raman Suri, Busi Siddhardha, and Senthil Prasad Elumalai. Ion-pumping microbial rhodopsin protein classification by machine learning approach. *BMC bioinformatics*, 24(1):29, 2023.

[40] John Shawe-Taylor and Nello Cristianini. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[41] Hiroshi Shimodaira, Ken-ichi Noma, Mitsuru Nakai, and Shigeki Sagayama. Dynamic time-alignment kernel in support vector machine. *Advances in neural information processing systems*, 14, 2001.

[42] Alex J Smola and Bernhard Schölkopf. *Learning with kernels*, volume 4. Citeseer, 1998.

[43] Ingo Steinwart. On the influence of the kernel on the consistency of support vector machines. *Journal of machine learning research*, 2(Nov):67–93, 2001.

[44] Anthony Thomas, Sanjoy Dasgupta, and Tajana Rosing. A theoretical perspective on hyperdimensional computing. *Journal of Artificial Intelligence Research*, 72:215–249, 2021.

[45] Nikil Wale, Ian A Watson, and George Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*, 14:347–375, 2008.

[46] Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. *Advances in neural information processing systems*, 13, 2000.

[47] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[48] Weihong Xu, Jaeyoung Kang, and Tajana Rosing. Fsl-hd: Accelerating few-shot learning on reram using hyperdimensional computing. In *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6. IEEE, 2023.

[49] Tao Yu, Yichi Zhang, Zhiru Zhang, and Christopher De Sa. Understanding hyperdimensional computing for parallel single-pass learning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[50] Tianqi Zhang, Justing Morris, Kenneth Stewart, Hin Wai Lui, Behnam Khaleghi, Anthony Thomas, Thiago Goncalves-Marback, Baris Aksanli, Emre O Neftci, and Tajana Rosing. : Accelerating event-based workloads with hyperdimensional computing and spiking neural networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2023.

[51] Quanling Zhao, Kai Lee, Jeffrey Liu, Muhammad Huzaifa, Xiaofan Yu, and Tajana Rosing. Fedhd: federated learning with hyperdimensional computing. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, pages 791–793, 2022.