Journal of Artificial Intelligence Research 82 (2025) 1293-1323 Submitted 11/2024; published 03/2025

Laplace-HDC: Understanding the Geometry of Binary Hyperdimensional Computing

Saeid Pourmand POURMANS@OREGONSTATE.EDU
School of Electrical Engineering and Computer Science
Oregon State University

Wyatt D. Whiting WHITINWY@OREGONSTATE.EDU
(Corresponding author)
Department of Mathematics, Oregon State University

Alireza Aghasi ALIREZA.AGHASI@OREGONSTATE.EDU
School of Electrical Engineering and Computer Science
Oregon State University

Nicholas F. Marshall MARSNICH@OREGONSTATE.EDU
Department of Mathematics, Oregon State University

Abstract

This paper studies the geometry of binary hyperdimensional computing (HDC), a computational scheme in which data are encoded using high-dimensional binary vectors. We establish a result about the similarity structure induced by the HDC binding operator and show that the Laplace kernel naturally arises in this setting, motivating our new encoding method *Laplace-HDC*, which improves upon previous methods. We describe how our results indicate limitations of binary HDC in encoding spatial information from images and discuss potential solutions, including using Haar convolutional features and the definition of a translation-equivariant HDC encoding. Several numerical experiments highlighting the improved accuracy of Laplace-HDC in contrast to alternative methods are presented. We also numerically study other aspects of the proposed framework, such as robustness and the underlying translation-equivariant encoding.

1. Introduction

Hyperdimensional computing (HDC) is a computational paradigm rooted in cognitive science and inspired by the operation of the brain. With billions of neurons and trillions of synapses, the human brain exhibits states akin to high-dimensional arrays. Unlike conventional machine learning models that work with floating-point operations, the brain's processes engage in simpler "arithmetic" but across significantly higher dimensions (such as the operations in the cerebral cortex). HDC aims to mimic the brain's operation by encoding data with high-dimensional vectors, called hypervectors, while using simple operations, such as the XOR operation. Hypervectors are often defined randomly or pseudo-randomly and can have entries that are binary, integer, real, or complex (Kleyko et al., 2023b); however, in this paper, we restrict our attention to binary HDC models, which are the most common in practice. In contrast with typical floating-point operations on data, the simplicity of binary operations makes binary HDC computationally straightforward and amenable to hardware-level optimization (Hernández-Cano et al., 2021a).

Similar to the cognitive operations of the brain, HDC is robust to noise, heavily distributable, interpretable, and energy efficient (Kanerva, 2009; Thomas et al., 2021). Additionally, HDC models can undergo single-pass training, where a model is trained by processing each sample in the data set only once. The simplicity of arithmetic tasks and their parallel nature facilitate rapid inference for these models. Thanks to these attributes, HDC is a well-suited framework for the Internet of Things (IoT) and edge devices (Khaleghi et al., 2021; Kleyko et al., 2023a), where resilience to noise and straightforward computations hold significant importance. Despite the simple underlying arithmetic, HDC models are considered across various complex tasks, such as speech recognition (Imani et al., 2017), written language classification (Karunaratne et al., 2021), DNA pattern matching (Kim et al., 2020), robotics (Neubert et al., 2019), image description tasks (Neubert & Schubert, 2021; Schlegel et al., 2022), and low energy computing (Basaklar et al., 2021; Chuang et al., 2020; Imani et al., 2019; Karunaratne et al., 2020).

A significant challenge associated with HDC models is their relatively low accuracy. For instance, as demonstrated in the experiments section below, standard HDC models attain an average accuracy rate of 82% on the MNIST handwritten digit classification task, while several variants of deep neural networks can achieve accuracies above 99.5% (Byerly et al., 2021; Hirata & Takahashi, 2023; Kabir et al., 2023). Standard HDC modeling consists of three main steps: hyperdimensional embedding, single-pass training, and inference. Efforts to enhance the accuracy of these models typically involve either refining one of these steps or proposing additional steps in the pipeline. For example, a standard HDC model uses a record-based encoding, which involves the binding and bundling of random hypervectors associated with the feature positions and feature values, while to boost the accuracy for temporal or spatial data, an N-gram based encoding may be considered, where feature positions are encoded into the hypervectors through rotational permutations (Ge & Parhi, 2020). As an example of changing the number of steps in the process, OnlineHD (Hernández-Cano et al., 2021a) is an HDC framework where an additional retraining phase is integrated into the modeling pipeline. This augmentation boosts the model accuracy by discarding the mispredicted queries from the corresponding mispredicted classes and adding them to the correct class.

1.1 Motivation

In this paper, we are primarily motivated by the work of Yu *et al.* (2022), which, in contrast to previous works that mainly focused on using either deterministic hypervectors or random hypervectors with i.i.d. entries, considers constructions of random hypervectors with a prescribed covariance structure. They refer to their approach as RFF-HDC and empirically show that it outperforms previous HDC schemes. More precisely, given a desired covariance structure $\boldsymbol{K} \in \mathbb{R}^{m\times m}$, Yu *et al.* (2022) uses the following algorithm to construct a matrix $\boldsymbol{V} \in \{-1,+1\}^{N\times m}$ of m hypervectors.

Require: Similarity matrix $\boldsymbol{K} \in \mathbb{R}^{n \times n}$, hyperdimension N
$\boldsymbol{W} \leftarrow \sin\left(\frac{\pi}{2}\boldsymbol{K}\right)$ (where the function sin is applied entrywise)
$\boldsymbol{U}\boldsymbol{S}\boldsymbol{U}^T \leftarrow \boldsymbol{W}$ (eigendecomposition)
Generate $\boldsymbol{G} \in \mathbb{R}^{N \times m}$ with i.i.d. standard Gaussian entries
$\boldsymbol{V} \leftarrow \operatorname{sign}(\boldsymbol{G}\boldsymbol{S}_+^{1/2}\boldsymbol{U})$ (with sign applied entrywise and $\boldsymbol{S}_+$ sets negative entries to 0).
return $\boldsymbol{V} \in \{-1, +1\}^{N \times m}$

When $\boldsymbol{S} = \boldsymbol{S}_+$, that is, when $\boldsymbol{W}$ is positive semi-definite, it follows that $\boldsymbol{V}^\top \boldsymbol{V}/N = \boldsymbol{K}$, see §1.4 for a more precise statement and mathematical description. The assumption that $\boldsymbol{W}$ is positive semi-definite constrains the types of similarities that can be achieved using this algorithm. However, Yu *et al.* (2022) shows that some restriction is necessary by proving that there are covariance structures $\boldsymbol{K}$ which are impossible to realize using binary hypervectors.

The current paper builds upon Yu *et al.* (2022) by studying the geometry of HDC encodings resulting from constructions using hypervectors with a covariance structure. More precisely, we consider the similarity structure of the embedding space induced by the HDC binding operation, see (1). We show that the Laplace kernel naturally arises in this context, and our results provide heuristics for choosing effective distributions of hypervectors. Moreover, we consider several modifications to the HDC pipeline, which further boost the accuracy of HDC models. We demonstrate theoretically and empirically how spatial information for images is lost in the similarity structure of certain HDC encoding schemes and present methods of retaining this information, including the definition of a translation-equivariant HDC encoding scheme. In addition to conducting empirical experiments to assess the proposed framework's performance compared to state-of-the-art techniques, mathematical tools are used to explore theoretical aspects.

We emphasize that some models we explore (as with previous work) involve using floating-point operations during the construction of the hypervectors or training stages of the models. We will emphasize when this is the case. Moreover, as we will discuss, these models have variants that can ultimately be fully represented and operated in a binary mode for inference, and results for binary inference will be presented.

1.2 Preliminaries and Notation

While implementations of binary HDC use vectors $\boldsymbol{x} \in \{0, 1\}^N$ (for some large N on the order of $N = 10^4$) equipped with entrywise XOR (denoted $\oplus$), we may conceptualize these vectors as being in $\{-1, +1\}^N$ equipped with entrywise product (denoted $\odot$). These two representations are isomorphic: if $\phi : \{0, 1\}^N \to \{-1, +1\}^N$ by $\boldsymbol{x} \mapsto \phi(\boldsymbol{x}) = \mathbf{1} - 2\boldsymbol{x}$ then

$$\phi(\boldsymbol{x}(i) \oplus \boldsymbol{y}(i)) = 1 - 2(\boldsymbol{x}(i) \oplus \boldsymbol{y}(i)) = \phi(\boldsymbol{x}(i)) \odot \phi(\boldsymbol{y}(i)),$$

for all $i = 1, \ldots, N$, where $\boldsymbol{x}(i)$ denotes the i-th entry of $\boldsymbol{x}$. In this paper, we use the $(\{-1, +1\}^N, \odot)$ representation of binary HDC schemes.

For a given hypervector $\boldsymbol{u} \in \{-1, +1\}^N$, we denote its k-th entry by $\boldsymbol{u}(k)$. We write $\boldsymbol{e}_k$ to denote the k-th standard basis vector whose k-th entry is equal to 1 and which is zero elsewhere. We use the convention that hypervectors are column vectors such that the inner

product can be expressed by

$$\boldsymbol{u}^\top \boldsymbol{v} = \sum_{k=1}^{N} \boldsymbol{u}(k)\boldsymbol{v}(k),$$

where $\boldsymbol{u}^\top$ denotes the transpose of $\boldsymbol{u}$, and write $\boldsymbol{u} \odot \boldsymbol{v}$ to denote the entrywise product

$$(\boldsymbol{u} \odot \boldsymbol{v})(k) = \boldsymbol{u}(k)\boldsymbol{v}(k),$$

for $k = 1, \ldots, N$. Given a set of d hypervectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d$, let

$$\bigodot_{j=1}^{d} \boldsymbol{v}_j = \boldsymbol{v}_1 \odot \cdots \odot \boldsymbol{v}_d,$$

denote the entrywise product over all vectors in the set. For a matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$, we write $\|\boldsymbol{A}\|_0$ to denote the number of nonzero entries of $\boldsymbol{A}$

$$\|\boldsymbol{A}\|_0 := \#\{(i,j) \in \{1, \ldots, N\}^2 : \boldsymbol{A}(i,j) \neq 0\},$$

where $\boldsymbol{A}(i,j)$ denotes the (i,j)-th entry of $\boldsymbol{A}$, and $\#$ denotes the counting measure.

1.3 Defining an Embedding

Assume that data $\mathcal{X} \subset \{1, \ldots, m\}^d$ are given, that is, each $\boldsymbol{x} \in \mathcal{X}$ is a d-dimensional vector whose entries are integers in the set $\{1, \ldots, m\}$. Let m hypervectors

$$\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m \in \{-1, +1\}^N,$$

be given (see §1.4 for a discussion of constructing these hypervectors). Further, let $\mathcal{P}$ be a set of d permutation matrices of size $N \times N$,

$$\mathcal{P} = \{\boldsymbol{\Pi}_1, \boldsymbol{\Pi}_2, \ldots, \boldsymbol{\Pi}_d\};$$

examples of families of permutation matrices of interest are discussed below. The binding operation which maps $\mathcal{X} \to \{-1, +1\}^N$ is defined by

$$\boldsymbol{x} \mapsto \boldsymbol{\psi}_{\boldsymbol{x}} = \bigodot_{i=1}^{d} \boldsymbol{\Pi}_i \boldsymbol{v}_{\boldsymbol{x}(i)}. \tag{1}$$

For the encoding $\boldsymbol{\psi}_{\boldsymbol{x}}$ to be meaningful, some assumptions must be imposed on the permutation matrices. We make the following trace-orthogonality assumption.

Assumption 1.1 (Trace-orthogonal family of permutations)**.** *We say that a family of permutations $\mathcal{P} = \{\boldsymbol{\Pi}_1, \boldsymbol{\Pi}_2, \ldots, \boldsymbol{\Pi}_d\}$ is trace-orthogonal if*

$$\langle \boldsymbol{\Pi}_i, \boldsymbol{\Pi}_{i'} \rangle = \operatorname{Tr}\left(\boldsymbol{\Pi}_i^\top \boldsymbol{\Pi}_{i'}\right) = 0, \qquad \forall i, i' \in \{1, \ldots, d\},\ i \neq i'. \tag{2}$$

It is straightforward to construct a family of trace-orthogonal permutations using cyclic shifts (under the necessary assumption that $d \leq N$).

Remark 1.1 (1D-Cyclic family). For $i \in \{1, \dots, d\}$ let $\boldsymbol{T}_i^{\text{1D-Cyclic}}$ denote the $N \times N$ permutation matrix, which acts on $\boldsymbol{v} \in \{-1, +1\}^N$ by

$$(\boldsymbol{T}_i^{\text{1D-Cyclic}} \boldsymbol{v})(i') = \boldsymbol{v}(i + i'), \quad \text{for} \quad i' \in \{1, \dots, N\}, \tag{3}$$

where the addition $i + i'$ is taken modulo N. That is, $\boldsymbol{T}_i^{\text{1D-Cyclic}}$ can be defined entrywise by

$$\boldsymbol{T}_i^{\text{1D-Cyclic}}(j, j') = \begin{cases} 1 & \text{if } j = j' + i \mod N \\ 0 & \text{otherwise.} \end{cases}$$

When $i \neq i'$ and $d \leq N$, the support of $\boldsymbol{T}_i^{\text{1D-Cyclic}}$ and $\boldsymbol{T}_{i'}^{\text{1D-Cyclic}}$ are disjoint so the trace-orthogonal property holds.

Remark 1.2 (1D-Block Cyclic family). Suppose that $N = dM$ for some positive integer M. Then, another family of trace-orthogonal permutations matrices $\{\boldsymbol{T}_i^{\text{1D-Block}} : i \in \{1, \dots, d\}\}$ can be defined by their action on $\boldsymbol{v} \in \{-1, +1\}^{d \times M}$ by

$$(\boldsymbol{T}_i^{\text{1D-Block}} \boldsymbol{v})(i', k) = \boldsymbol{v}(i + i', k), \quad \text{for} \quad (i', k) \in \{1, \dots, d\} \times \{1, \dots, M\}, \tag{4}$$

where the addition $i + i'$ is taken modulo d. It is straightforward to verify that this 1D-Block Cyclic family is also trace-orthogonal.

1.4 Constructing Hypervectors

We choose hypervectors using the method of Yu *et al.* (Yu et al., 2022) outlined in §1.1 above. In the following, we describe this construction in detail and provide related mathematical preliminaries. Given an affinity matrix $\boldsymbol{K} \in \mathbb{R}^{m \times m}$, the goal is to construct hypervectors $\boldsymbol{v}_1, \dots, \boldsymbol{v}_m \in \{-1, +1\}^N$ such that

$$\mathbb{E} \frac{\boldsymbol{v}_i^\top \boldsymbol{v}_j}{N} = \boldsymbol{K}(i, j).$$

Recall Grothendieck's identity (see, for example, Vershynin (2018, page 63)).

Lemma 1.1 (Grothendieck's identity)**.** *Let $\boldsymbol{g}$ be an n-dimensional vector with i.i.d. random standard Gaussian entries. Then, for any fixed vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{S}^{n-1}$, we have*

$$\mathbb{E}(\operatorname{sign}(\boldsymbol{g}^\top \boldsymbol{u}) \operatorname{sign}(\boldsymbol{g}^\top \boldsymbol{v})) = \frac{2}{\pi} \arcsin(\boldsymbol{u}^\top \boldsymbol{v}),$$

where $\mathbb{S}^{n-1} = \{x \in \mathbb{R}^n : \|x\|_2 = 1\}$.

Suppose that an affinity kernel matrix $\boldsymbol{K} \in \mathbb{R}^{m \times m}$ is given, and define $\boldsymbol{W} \in \mathbb{R}^{m \times m}$ by

$$\boldsymbol{W}(i, j) = \sin\left(\frac{\pi}{2} \boldsymbol{K}(i, j)\right), \tag{5}$$

for $i, j = 1, \dots, m$. The construction is effective when the following assumption, which restricts the possible choices of $\boldsymbol{K}$, is satisfied.

Assumption 1.2 (Admissible affinity kernel)**.** *We say that an affinity kernel $\boldsymbol{K} \in \mathbb{R}^{m \times m}$ is admissible if the matrix $\boldsymbol{W}$ defined by* (5) *is symmetric positive semi-definite and $\boldsymbol{K}(i, i) = 1$ for all $i = 1, \dots, m$.*

An example of a family of admissible affinity kernels is provided in Corollary 1.1; also see §1.2. For now, we proceed under the assumption that $\boldsymbol{W}$ is a symmetric positive semi-definite matrix, which implies $\boldsymbol{W}$ can be decomposed as

$$\boldsymbol{W} = \boldsymbol{U}^\top \boldsymbol{U},$$

where $\boldsymbol{U}$ is a real-valued $n \times n$ matrix. Note that if $\boldsymbol{W}$ is not positive semi-definite, it is possible to truncate the negative eigenvalues to achieve a decomposition of this form, see §1.1; however, in this case, the construction will not achieve hypervectors with the covariance structure of $\boldsymbol{K}$.

Let $\boldsymbol{G} \in \mathbb{R}^{N\times m}$ be a matrix whose entries are independent Gaussian random variables with mean 0 and variance 1. Define the matrix $\boldsymbol{V} \in \{-1,+1\}^{N\times m}$ by

$$\boldsymbol{V} = \operatorname{sign}(\boldsymbol{G}\boldsymbol{U}),$$

where sign denotes the entrywise sign function with the convention that 0 has sign +1. Let $\boldsymbol{v}_k$ denote the k-th column of $\boldsymbol{V}$. Then, the hypervectors

$$\boldsymbol{v}_1, \dots, \boldsymbol{v}_m \in \{-1,+1\}^N,$$

have the desired covariance structure in expectation. Indeed, we claim that

$$\mathbb{E}\boldsymbol{v}_i(k)\boldsymbol{v}_j(k) = \boldsymbol{K}(i,j), \quad \text{for all} \quad k \in \{1,\dots,N\}, \tag{6}$$

which in turn implies

$$\mathbb{E}\frac{\boldsymbol{v}_i^\top \boldsymbol{v}_j}{N} = \boldsymbol{K}(i,j). \tag{7}$$

To show (6), we note that by the definition of $\boldsymbol{v}_j$ we have

$$\mathbb{E}\boldsymbol{v}_i(k)\boldsymbol{v}_j(k) = \mathbb{E}\operatorname{sign}(\boldsymbol{g}_k^\top \boldsymbol{u}_i)\operatorname{sign}(\boldsymbol{g}_k^\top \boldsymbol{u}_j),$$

where $\boldsymbol{g}_k^\top$ is the k-th row of $\boldsymbol{G}$ and $\boldsymbol{u}_j$ is the j-th column of $\boldsymbol{U}$. Recall that in the definition of an admissible kernel, we assume $\boldsymbol{K}(i,i) = 1$. It follows that

$$\|\boldsymbol{u}_i\|_2^2 = \boldsymbol{u}_i^\top \boldsymbol{u}_i = \boldsymbol{W}(i,i) = \sin\left(\frac{\pi}{2}\boldsymbol{K}(i,i)\right) = 1,$$

for $i = 1,\dots,m$. Thus, Grothendieck's Identity (see Lemma 1.1 above) can be applied to deduce that

$$\mathbb{E}\operatorname{sign}(\boldsymbol{g}_k^\top \boldsymbol{u}_i)\operatorname{sign}(\boldsymbol{g}_k^\top \boldsymbol{u}_j) = \frac{2}{\pi}\arcsin(\boldsymbol{u}_i^\top \boldsymbol{u}_j). \tag{8}$$

By the definition of $\boldsymbol{U}$ we have

$$\frac{2}{\pi}\arcsin(\boldsymbol{u}_i^\top \boldsymbol{u}_j) = \frac{2}{\pi}\arcsin\left(\boldsymbol{W}(i,j)\right) = \boldsymbol{K}(i,j),$$

where the final equality follows from the definition of $\boldsymbol{W}$.

Before stating our main theoretical result two remarks are in order.

Remark 1.3 (Intuition for kernel-based hypervectors sampling instead of uniform sampling). We emphasize that in contrast to some classic works on HDC that involve sampling hypervectors independently and uniformly from $\{-1,+1\}^N$, we use a kernel-based approach, motivated by (Yu et al., 2022), that creates correlated hypervectors. In particular, the hypervectors $\boldsymbol{v}_1,\dots,\boldsymbol{v}_k$ in our method are selected to have a prescribed inner-product similarity structure with one another, see (7). Earlier works have noted that hypervectors sampled uniformly at random from $\{-1,+1\}^N$ have a high probability of being nearly orthogonal, and this property is desirable when selecting vectors that will represent unrelated features in the encoded data (Kanerva, 2009). However, often the features present in the data will have a natural similarity structure. For example, greyscale colors have a natural similarity based on how similar the tones appear to the eye. In contrast, when features in the input data share no similarity and are invariant to relabeling, then using orthogonal vectors could be a natural choice.

Remark 1.4 (Theoretical and empirical justification of kernel-based hypervector sampling). Our theoretical and experiment results further demonstrate the utility of using kernel-based hypervector sampling. In particular, our theoretical results in §1.5 describes how the hypervector similarity structure determines the encoding similarity structure. Further, our reuslts in §1.6 determine conditions that a desired similarity kernel $\boldsymbol{K}(i,j)$ must satisfy to be viable using our method. In contrast, previous work focused mainly on heuristic constructions without precise theory. Our numerical results demonstrate that our Laplace-HDC method achieves state-of-the-art performance among HDC schemes, see §2.5, Table 5, which includes comparison to classical HDC that uses uniform sampling. Finally, we note that there are alternative heuristic approaches to improving HDC such as the record-based encoder used in the LeHDC method developed by Duan et al. (2022a), see §3 for a further discussion.

1.5 Main Analytic Result

Let data $\mathcal{X}\subset\{1,\dots,m\}^d$ be given, and fix a hyperdimension $N\geq d$. Let $\mathcal{P}=\{\boldsymbol{\Pi}_1,\boldsymbol{\Pi}_2,\dots,\boldsymbol{\Pi}_d\}$ be a family of $N\times N$ permutation matrices satisfying Assumption 1.1. Assume that $\boldsymbol{K}\in\mathbb{R}^{m\times m}$ is an affinity kernel which is admissible in the sense of Assumption 1.2. Construct $\boldsymbol{v}_1,\dots,\boldsymbol{v}_m\in\{-1,+1\}^N$ using $\mathcal{P}$ and $\boldsymbol{K}$ in the procedure described in §1.4. Define the embedding $\boldsymbol{\psi}_{\boldsymbol{x}}\in\{-1,+1\}^N$ of a vector $\boldsymbol{x}\in\mathcal{X}$ in terms of the binding operation

$$\boldsymbol{x}\mapsto\boldsymbol{\psi}_{\boldsymbol{x}}=\bigodot_{i=1}^{d}\boldsymbol{\Pi}_i\boldsymbol{v}_{\boldsymbol{x}(i)}. \tag{9}$$

The following theorem is our main analytic result.

Theorem 1.1. *Under the assumptions of §1.5, we have*

$$S(\boldsymbol{x},\boldsymbol{y}):=\mathbb{E}\frac{\boldsymbol{\psi}_{\boldsymbol{x}}^\top\boldsymbol{\psi}_{\boldsymbol{y}}}{N}=\prod_{i=1}^{d}\boldsymbol{K}(\boldsymbol{x}(i),\boldsymbol{y}(i)), \tag{10}$$

and

$$\operatorname{Var}\left(\frac{\boldsymbol{\psi}_{\boldsymbol{x}}^\top\boldsymbol{\psi}_{\boldsymbol{y}}}{N}\right)\leq\frac{2\gamma_{\mathcal{P}}}{N^2}(1-S(\boldsymbol{x},\boldsymbol{y})),$$

where for the set $\mathcal{P} = \{\mathbf{\Pi}_1, \mathbf{\Pi}_2, \ldots, \mathbf{\Pi}_d\}$*:*

$$\gamma_{\mathcal{P}} = \left\| \sum_{i'=1}^{d} \sum_{i=1}^{d} \mathbf{\Pi}_i \mathbf{\Pi}_{i'}^{\top} \right\|_0 . \tag{11}$$

The proof of Theorem 1.1 is given in §A.1. The following corollary states this result for the case where $\mathcal{P}$ is the 1D-Cyclic family of permutation matrices, and $\boldsymbol{K}$ is approximately the Laplace kernel. To simplify the exposition, we present an informal version of this result here and provide a more technical statement in the following section.

Corollary 1.1 (Informal Statement)**.** *Let* $\mathcal{P} = \{\mathbf{\Pi}_1, \ldots, \mathbf{\Pi}_d\}$ *be the 1D-Cyclic family of permutation matrices defined in Remark 1.1, which satisfies Assumption 1.1. It is possible to choose* $\boldsymbol{K}$ *approximately equal to the Laplace kernel* $\boldsymbol{K}(i,j) \approx \exp(-\lambda|i-j|)$ *such that Assumption 1.2 is satisfied. In this case,*

$$S(\boldsymbol{x}, \boldsymbol{y}) := \mathbb{E}\left(\frac{\boldsymbol{\psi}_{\boldsymbol{x}}^{\top} \boldsymbol{\psi}_{\boldsymbol{y}}}{N} \right) \approx \exp(-\lambda \|\boldsymbol{x} - \boldsymbol{y}\|_1). \tag{12}$$

Moreover, the variance satisfies

$$\mathrm{Var}\left(\frac{\boldsymbol{\psi}_{\boldsymbol{x}}^{\top} \boldsymbol{\psi}_{\boldsymbol{y}}}{N} \right) \leq \frac{4d-2}{N}(1 - S(\boldsymbol{x}, \boldsymbol{y})).$$

For a precise version of the corollary, see Theorem 1.2 below. Interestingly, it is not possible to replace the Laplace kernel with the Gaussian kernel in this statement, see §1.6

1.6 Choosing an Admissible Kernel

Recall that $\boldsymbol{K} \in \mathbb{R}^{m \times m}$ is admissible in the sense of Assumption 1.2 if $\boldsymbol{K}(i,i) = 1$ and if $\boldsymbol{W}$ defined by

$$\boldsymbol{W}(i,j) = \sin\left(\frac{\pi}{2} \boldsymbol{K}(i,j) \right)$$

is positive semi-definite. In §1.6.1, we conduct an informal study of admissible kernels. Subsequently, in §1.6.2, we state a precise result that describes a family of admissible kernels $\boldsymbol{K}_\alpha$ and the resulting expected similarity $\boldsymbol{S}_\alpha(\boldsymbol{x}, \boldsymbol{y})$.

1.6.1 Heuristic Derivation

Given an admissible kernel $\boldsymbol{K}$, Theorem 1.1 states that the resulting expected similarity $S(\boldsymbol{x}, \boldsymbol{y})$ has the form

$$S(\boldsymbol{x}, \boldsymbol{y}) = \prod_{i=1}^{d} \boldsymbol{K}(\boldsymbol{x}(i), \boldsymbol{y}(i)).$$

This formula suggests that the values of $\boldsymbol{K}(i,j)$ should all be relatively close to 1 for all i, j since, otherwise, taking the product will result in values close to zero; this observation motivates the ansatz

$$\boldsymbol{K}(i,j) = 1 - \boldsymbol{F}(i,j), \quad \text{where} \quad 0 \leq \boldsymbol{F}(i,j) \leq \varepsilon,$$

for some small $\varepsilon > 0$. The series expansions

$$\exp(x) = 1 + x + \mathcal{O}(x^2), \quad \text{and} \quad \sin\left(\frac{\pi}{2}(1-x)\right) = 1 - \frac{\pi^2 x^2}{8} + \mathcal{O}(x^4),$$

as $x \to 0$ imply that the resulting matrix $\boldsymbol{W}$ satisfies

$$\boldsymbol{W}(i,j) = \sin\left(\frac{\pi}{2}\left(1 - \boldsymbol{F}(i,j)\right)\right) = \exp\left(-\frac{\pi^2}{8}\boldsymbol{F}(i,j)^2\right) + \mathcal{O}(\varepsilon^4).$$

These calculations motivate choosing $\boldsymbol{F}(i,j)$ such that

$$\boldsymbol{W}(i,j) \approx \exp\left(-\frac{\pi^2}{8}\boldsymbol{F}(i,j)^2\right)$$

is positive semi-definite. One natural choice is setting $\boldsymbol{F}(i,j) = \lambda|i-j|$ such that $\boldsymbol{K}$ is approximately equal to the Laplace kernel $\boldsymbol{K}(i,j) \approx \exp(-\lambda|i-j|)$ and $\boldsymbol{W}$ is approximately equal to the Gaussian kernel

$$\boldsymbol{W}(i,j) \approx \exp\left(-|i-j|^2/(2\sigma^2)\right),$$

where $\sigma = 2\lambda/\pi$. Both the Laplace kernel and the Gaussian kernel are positive definite kernels, so the Laplace kernel $\boldsymbol{K}$ is one natural choice to use in this construction.

Interestingly, choosing $\boldsymbol{F}(i,j) = |i-j|^2/(2\sigma^2)$ such that $\boldsymbol{K}$ is approximately equal to the Gaussian kernel $\boldsymbol{K}(i,j) \approx \exp(-\lambda|i-j|^2/(2\sigma^2)$ results in $\boldsymbol{W}$ of the form

$$\boldsymbol{W}(i,j) \approx \exp\left(-\gamma|i-j|^4\right),$$

for $\gamma = \pi^2/(32\sigma^4)$, which is not a positive definite kernel. Thus, the Gaussian kernel is not an admissible choice of $\boldsymbol{K}$ for this construction.

1.6.2 Precise Description

In the following, we make the informal derivation of the previous section, which says the Laplace kernel is a natural choice for $\boldsymbol{K}$ more rigorous and general. In particular, we define a family of admissible kernels and derive the resulting expected similarity kernel.

Lemma 1.2. *Fix $\alpha, \lambda > 0$, and consider the matrix $\boldsymbol{W}_\alpha \in \mathbb{R}^{m\times m}$ with elements*

$$\boldsymbol{W}_\alpha(i,j) = \exp\left(-\lambda|i-j|^\alpha\right), \tag{13}$$

for all $i, j = 1, \ldots, m$. Then,

- *For each $\alpha \in [0,2]$, $\boldsymbol{W}_\alpha$ is positive semi-definite*
- *For each $\alpha \in (2,\infty)$, $\boldsymbol{W}_\alpha$ is not positive semi-definite*

Proof of Lemma 1.2. This is a direct implication of Schoenberg's classic result. Specifically, Corollary 3 of Schoenberg (1938) states that $\exp\left(-|x|^\alpha\right)$ is positive definite when $0 < \alpha \leq 2$, and fails to become positive definite when $\alpha > 2$. This function being positive definite implies that for any selection of $x_1, \ldots, x_m \in \mathbb{R}$, the matrix with elements $\exp\left(-|x_i - x_j|^\alpha\right)$ is positive semi-definite. Narrowing down the choice of the test points to $x_i = \lambda^{1/\alpha} i$ for $i = 1, \ldots, m$ validates the claims of the lemma. □

The following theorem defines a family of admissible kernels and, motivated by the informal derivation of §1.6.1, derives the resulting expected similarity kernel. A subsequent remark describes the connection to the Laplace kernel derived in the previous section.

Theorem 1.2. *Let $\lambda > 0$ and $\alpha \in (0, 1]$. Define $\boldsymbol{K}_\alpha \in \mathbb{R}^{m\times m}$ by*

$$\boldsymbol{K}_\alpha(i,j) = \frac{2}{\pi}\arcsin\left(\exp\left(-\frac{\pi^2}{8}\lambda|i-j|^{2\alpha}\right)\right), \tag{14}$$

for $i, j \in \{1, \dots, m\}$. Then, $\boldsymbol{K}_\alpha$ is admissible in the sense of Assumption 1.2. Moreover, if the bandwidth parameter λ is set such that $\lambda|i-j|^\alpha \le \varepsilon$, then

$$\boldsymbol{K}_\alpha(i,j) = \exp\left(-\lambda|i-j|^\alpha\right) + \mathcal{O}(\varepsilon^2),$$

and the resulting expected similarity kernel S_α satisfies

$$S_\alpha(\boldsymbol{x},\boldsymbol{y}) = \exp(-\lambda\|\boldsymbol{x}-\boldsymbol{y}\|_\alpha^\alpha)(1+\mathcal{O}(\varepsilon^2 d)),$$

as $\varepsilon \to 0$.

The proof of Theorem 1.2 is given in §A.2. In the following, we make three remarks about this result.

Remark 1.5 (Laplace kernel). When $\alpha = 1$

$$\boldsymbol{K}_\alpha(i,j) = \exp(-\lambda|i-j|) + \mathcal{O}(\varepsilon^2),$$

is the Laplace kernel, up to lower order terms, and

$$\boldsymbol{S}_\alpha(i,j) = \exp(-\lambda\|\boldsymbol{x}-\boldsymbol{y}\|_1)(1+\mathcal{O}(\varepsilon^2 d)),$$

which recovers the informal motivating result from the previous section.

Remark 1.6 (Limitation of similarity structure). The result of Theorem 1.1 suggests a limitation of binary HDC pipelines that involve binding data $\boldsymbol{x} \mapsto \boldsymbol{\psi}_{\boldsymbol{x}}$ and then relying on the similarity structure induced by inner products $\boldsymbol{\psi}_{\boldsymbol{x}}^\top \boldsymbol{\psi}_{\boldsymbol{y}}/N$. Namely, the expected similarity $S(\boldsymbol{x},\boldsymbol{y})$ is invariant to global permutations of the elements of the data, that is,

$$S(\boldsymbol{x},\boldsymbol{y}) = S(\boldsymbol{x}\circ\sigma, \boldsymbol{y}\circ\sigma),$$

where $(\boldsymbol{x}\circ\sigma)(i) = \boldsymbol{x}(\sigma(i))$ for a permutation σ, see (10). For some data types, such as images, the ordering of the data elements may contain information. For example, a group of black pixels may indicate structure, such as a digit, while scattered black pixels may be noise. These spatial relationships can be captured by using feature extraction methods such as convolutional filters, see §2.4. Alternatively, the definition of the embedding could be modified so that spatial information is encoded via another mechanism such as translation-equivariance, see §1.7

Remark 1.7 (Setting the bandwidth parameter λ). When $\boldsymbol{K}_\alpha$ is defined by (14), the form of the resulting expected similarity

$$S_\alpha(\boldsymbol{x}, \boldsymbol{y}) \approx \exp(-\lambda \|\boldsymbol{x} - \boldsymbol{y}\|_\alpha^\alpha),$$

can be used to set the bandwidth parameter λ. Suppose a data set $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^n$ is given. A typical strategy for kernel methods is setting the bandwidth parameter so that $\boldsymbol{K}_\alpha$ is invariant to transformations that preserve distances $\|\boldsymbol{x} - \boldsymbol{y}\|_\alpha$ up to a global constant (for example, global scaling of the data). One way to achieve this is by setting

$$\lambda = \frac{c}{\text{median}(\boldsymbol{D})}, \tag{15}$$

for some constant $c > 0$, where

$$\boldsymbol{D}(i, j) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_\alpha^\alpha,$$

and median($\boldsymbol{D}$) is the median of the n^2 numbers in the matrix $\boldsymbol{D}$. When the number of data points n is large, the matrix $\boldsymbol{D}$ in (15) could be replaced by a matrix $\boldsymbol{D}'$ corresponding to a subset of the data to reduce the required computation. The hyperparameter c in any given classification task can be determined via cross-validation or hold-out methods depending on the size of the input data. In §2.2 we discuss the choice of this hyper-parameter in our image classification problem.

1.7 Translation Equivariant Encoding

Remark 1.6 describes a limitation of the method related to encoding spatial information. In the following, we describe an alternate way to encode spatial information by defining a translation equivariant binding operation for images. Suppose that $\boldsymbol{x} \in \{1, \ldots, m\}^{L \times L}$ is an $L \times L$ image. For simplicity, initially assume $N := L^2$ (We will subsequently relax this assumption to handle $N > L^2$). We define a family of permutations $\mathcal{P} = \{\boldsymbol{T}_{i,j} : i, j = 1, ..., m\}$, which act on $\boldsymbol{v} \in \{-1, +1\}^{L \times L}$ by

$$(\boldsymbol{T}_{i,j}\boldsymbol{v})(i', j') = \boldsymbol{v}(i + i', j + j'),$$

where the addition $i + i'$ and $j + j'$ is taken modulo L. This family satisfies the Trace-Orthogonality Assumption 1.1. Define the embedding

$$\boldsymbol{x} \mapsto \boldsymbol{\psi}_{\boldsymbol{x}} = \bigodot_{i,j=1}^{L} \boldsymbol{T}_{i,j}\boldsymbol{v}_{\boldsymbol{x}(i,j)}, \tag{16}$$

where $\boldsymbol{x}(i, j)$ denotes the (i, j)-th entry of $\boldsymbol{x}$. By construction, this embedding is translation-equivariant. That is,

$$\boldsymbol{T}_{i,j}\boldsymbol{\psi}_{\boldsymbol{x}} = \boldsymbol{\psi}_{\boldsymbol{T}_{i,j}\boldsymbol{x}},$$

see Figure 1.

The equivariant binding operation for the case $N = L^2$ can be extended to $N > L^2$ while maintaining exact translation-equivariance by using a block-based construction when

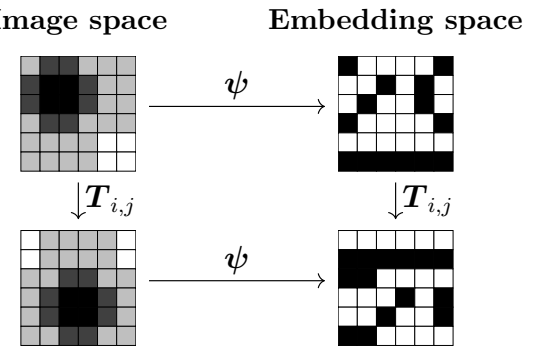

Figure 1: Communitive diagram for tanslation-equivariance of binding operation (16).

$N = ML^2$ by taking M copies of the $L \times L$ construction. More precisely, we define the 2D-Block family of permutations matrices $\{\boldsymbol{T}_{i,j}^{\text{2D-Block}} : i,j \in \{1,\dots,L\}\}$ which act on $\boldsymbol{v} \in \{-1,+1\}^{L\times L\times M}$ by

$$\left(\boldsymbol{T}_{i,j}^{\text{2D-Block}}\boldsymbol{v}\right)(i',j',k) = \boldsymbol{v}(i+i',j+j',k), \quad \text{for} \quad (i',j',k) \in \{1,\dots,L\}^2\times\{1,\dots,M\}, \tag{17}$$

where the addition $i+i'$ and $j+j'$ is taken modulo L. Alternatively, one can consider $N = M^2 > L^2$ and define $\{\boldsymbol{T}_{i,j}^{\text{2D-Cyclic}} : i,j \in \{1,\dots,L\}\}$

$$\left(\boldsymbol{T}_{i,j}^{\text{2D-Cyclic}}\boldsymbol{v}\right)(i',j') = \boldsymbol{v}(i+i',j+j'), \quad \text{for } (i',j') \in \{1,\dots,M\}^2, \tag{18}$$

where addition $i+i'$ and $j+j'$ is taken modulo M. When the 2D-Cyclic family of permutations is used in the binding operation, the resulting encoding is translation-equivariant when ignoring boundary effects. We demonstrate an application of the 2D-Cyclic family of permutations to visualize this equivariance property §2.7.

Remark 1.8 (Connections of constructions to neural networks). We note two connections of the binary HDC schemes we consider to neural networks. First, the Laplace kernel, which naturally arose in our binary HDC construction, also has connections to Neural Tangent Kernels (NTKs) (Geifman et al., 2020; Chen & Xu, 2020). Second, we note that neural networks with $+1$ and -1 weights and activations have been considered by several authors; see, for example, Courbariaux *et al.* (2016) and Kim (2016).

2. Experiments

We present several numerical experiments on the binary HDC schemes described in this paper and their limitations and extensions. Code for all presented methods is publically available at:

https://github.com/HDStat/Laplace-HDC

This section is organized as follows. In §2.1, we describe the linear classifiers that we use on the binary encodings. In §2.2, we present results for the vanilla version of Laplace-HDC. In §2.3, we present results for Laplace-HDC using singular value decomposition (SVD) features.

In §2.4, we present results for Laplace-HDC with Haar convolutional features. In §2.5, we provide a comparison of Laplace-HDC to other methods. Beyond the model accuracy, we study robustness in §2.6, and conduct experiments concerning the translation-equivariance property of the proposed encoding scheme in §2.7.

2.1 Classification Methods

Once the data $\boldsymbol{x} \in \mathcal{X}$ has been embedded $\boldsymbol{x} \mapsto \boldsymbol{\psi}_{\boldsymbol{x}}$, a classifier may be trained. Each classifier we consider uses an inner product to determine the final class. Suppose the classes $\{1, 2, \ldots, c\}$ are represented within $\mathcal{X}$, and suppose $\boldsymbol{\psi}_i$ denotes the class representative for class i, for $i = 1, 2, \ldots, c$. We determine the class of $\boldsymbol{y}$ by a simple linear classifier

$$\text{class}(\boldsymbol{y}) = \underset{i=1,\ldots,c}{\arg\max} \left(\boldsymbol{\psi}_{\boldsymbol{y}}^{\top} \boldsymbol{\psi}_i \right). \tag{19}$$

Below, we detail four methods we use for determining the class representatives $\{\boldsymbol{\psi}_i\}$.

2.1.1 Float and Binary Majority Vote

The majority vote classifiers operate on a simple principle for classification. First, we describe the binary flavor of this method. Consider the collection

$$C_i = \{\boldsymbol{x} \in \mathcal{X} : \text{class}(\boldsymbol{x}) = i\},$$

and let $\#(C_i)$ denote the number of elements in the collection. The representative for class i, denoted $\boldsymbol{\psi}_i$, is determined by a majority vote of $\boldsymbol{\psi}_{\boldsymbol{x}}$ for $\boldsymbol{x} \in C_i$. More precisely, we define the Binary Majority Vote representative $\boldsymbol{\psi}_i$ by

$$\boldsymbol{\psi}_i(k) = \begin{cases} +1 & \sum_{\boldsymbol{x} \in C_i} \boldsymbol{\psi}_{\boldsymbol{x}}(k) > 0 \\ -1 & \text{otherwise,} \end{cases}$$

The Float Majority Vote is largely the same, but we relax the condition that $\boldsymbol{\psi}_i$ must have entries in $-1, +1$. Instead, the entries of $\boldsymbol{\psi}_i$ may be floating-point numbers. In this case, the representative $\boldsymbol{\psi}_i$ is determined entrywise by class averages

$$\boldsymbol{\psi}_i(k) = \frac{1}{\#(C_i)} \sum_{\boldsymbol{x} \in C_i} \boldsymbol{\psi}_{\boldsymbol{x}}(k).$$

2.1.2 Float and Binary SGD

We define two classifiers, which we call Float SGD and Binary SGD. The Float SGD classifier determines the class representatives $\boldsymbol{\psi}_i$ by optimizing a cross-entropy loss function using stochastic gradient descent; more precisely, we use Adam (Kingma & Ba, 2017) with a learning rate parameter $\alpha = 0.01$, where the model takes $\boldsymbol{\psi}_{\boldsymbol{x}}$ and outputs one of the c classes. We perform 3 epochs training passes over the data in $\mathcal{X}$ in all experiments. The Binary HDC classifier operates in the same manner, except during each training epoch, the weights of the model parameter are clamped to be in the range $[-1, 1]$. Once all training epochs have been completed; the sign function is applied to the model weights, making all negative weights -1 and all positive weights $+1$.

2.2 Laplace-HDC in its Basic Form

In this section, we define a Binary HDC scheme motivated by Theorem 1.1, Corollary 1.1, and Theorem 1.2, which we call Laplace-HDC. Given a data set $\mathcal{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$ where $\boldsymbol{x}_i \in \{1, \dots, m\}^d$, Laplace-HDC consists of the following steps:

1. Set the bandwidth parameter $\lambda > 0$ using Remark 1.7 with $\alpha = 1$.
2. Define the kernel $\boldsymbol{K} \in \mathbb{R}^{m \times m}$ using (14).
3. Construct hypervectors $\boldsymbol{v}_1, \dots, \boldsymbol{v}_m \in \{-1, +1\}^N$ using the method detailed in §1.4.
4. Choose a family of permutation matrices $\mathcal{P} = \{\boldsymbol{\Pi}_1, \dots, \boldsymbol{\Pi}_d\}$ that are Trace-Orthogonal in the sense of Assumption 1.1.
5. Encode each $\boldsymbol{x}_j \mapsto \boldsymbol{\psi}_{\boldsymbol{x}_j}$ using the binding operation (9).
6. Return $\{\boldsymbol{\psi}_{\boldsymbol{x}_1}, \dots, \boldsymbol{\psi}_{\boldsymbol{x}_n}\}$ where $\boldsymbol{\psi}_{\boldsymbol{x}_i} \in \{-1, +1\}^N$.

By construction, it follows from Theorem 1.2 that

$$\mathbb{E}\frac{\boldsymbol{\psi}_{\boldsymbol{x}_i}^\top \boldsymbol{\psi}_{\boldsymbol{x}_j}}{N} \approx \exp(-\lambda \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_1).$$

In the following, we demonstrate the utility of Laplace-HDC in an application to image classification. We consider four choices of trace-orthogonal families of permutation matrices: 1D-Cyclic shift, 1D-Block Cyclic, 2D-Block Cyclic, and 2D-Cyclic Shift, which are defined in (3), (4), (17), and (18), respectively. Classification is performed by determining which class representative $\boldsymbol{\psi}_i$ gives the largest inner product (19). The class representatives $\boldsymbol{\psi}_i$ are determined by two different methods: Float SGD and Binary SGD, which are defined in §2.1.2.

In each case, we use the largest possible hyperdimensionality $N \leq 10^4$. For example, the 1D-Block family of permutations requires that $N = dM$ for some positive integer M. The image data we consider has dimension $d = 28^2$, so we choose $M = \lfloor 10^4/28^2 \rfloor = 12$, which results in $N = 9408$. When setting the bandwidth parameter $\lambda > 0$, we use (15) with $c = 1$ except for binary SGD where $c = 4$ provides better performance, and we estimate the median of the ℓ_1-distances of the data using 1000 samples selected uniformly at random from $\mathcal{X}$. The accuracy for each is presented as the mean $\pm$ one standard deviation in all cases; see Table 1.

Table 1: Basic Laplace-HDC Performance

	1D-Cyclic	2D-Cyclic	1D-Block	2D-Block
Float SGD - Fashion MNIST	86.06 ± 0.74	86.03 ± 0.73	87.86 ± 0.40	87.41 ± 0.15
Binary SGD - Fashion MNIST	83.60 ± 1.35	83.87 ± 0.63	84.72 ± 0.23	83.51 ± 0.57
Float SGD - MNIST	95.46 ± 0.48	95.46 ± 0.51	96.13 ± 0.25	96.14 ± 0.28
Binary SGD - MNIST	93.25 ± 1.09	93.37 ± 1.00	94.43 ± 0.82	94.59 ± 0.44

2.3 Laplace-HDC with SVD Features

Singular value decomposition is a popular pre-processing tool in predictive tasks involving numerical features. Well-known techniques such as principal component regression use this decomposition to rotate the coordinate system so that the features are uncorrelated in the rotated system. The process normally involves populating the numerical features into a matrix $\boldsymbol{X}$ (where the number of columns corresponds to the number of features and the number of rows corresponds to the number of samples) and then performing a compact SVD of the form $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ to acquire the transformation matrix. The new feature matrix takes the form $\hat{\boldsymbol{X}} = \boldsymbol{X}\boldsymbol{V}$, which not only enjoys uncorrelated features, also may have fewer columns than those in $\boldsymbol{X}$ thanks to the compact nature of the SVD operation (instead of a full SVD). Truncating small singular values is a manual alternative to reduce the number of features, which is also capable of denoising the feature matrix. This technique is incorporated into the HDC pipeline by first mapping the data matrix $\boldsymbol{X}$ to $\hat{\boldsymbol{X}}$ and then performing the hyperdimensional encoding on $\hat{\boldsymbol{X}}$.

In this section, we run some experiments to highlight the effect of having uncorrelated or loosely correlated features when using an HDC framework. We note that, in some cases, large datasets may have this property without needing a decorrelation process such as SVD. To evaluate the effects of SVD transformation, we applied the method to the MNIST and FashionMNIST data and used a total of 8 variations of permutation schemes and classifier pairs. The mean test accuracy (in percentage) of different classifiers, plus and minus one standard deviation, is computed for 20 independent experiments after SVD preprocessing and different permutation schemes. The results are available in Tables 2 and 3. The hyperparameters λ and N are set in the same way described in §2.2.

Furthermore, to avoid running an exact SVD in the original feature domain, we employ the randomized SVD technique proposed in (Halko, Martinsson, & Tropp, 2011). In these experiments, the decorrelation is performed alongside feature reduction (reducing the number of features to $d = 100$ or $d = 200$). Clearly, because of using an approximate SVD, the features are not expected to be fully decorrelated but rather loosely correlated. However, as summarized in Tables 2 and 3, the accuracies obtained are higher than working with raw features, showing a promise in the proposed framework.

Table 2: 1D-Cyclic Shift Laplace-HDC with exact and approximate SVD features

	Exact	Approx $d = 100$	Approx $d = 200$
Float - Fashion MNIST	84.437 ± 0.006	85.439 ± 0.002	85.751 ± 0.002
Binary - Fashion MNIST	83.946 ± 0.01	83.856 ± 0.01	84.811 ± 0.02
Float - MNIST	95.231 ± 0.004	96.632 ± 0.001	95.929 ± 0.001
Binary - MNIST	92.925 ± 0.009	$93.563 \pm 0.0.02$	94.349 ± 0.009

2.4 Laplace-HDC with Haar Convolutional Features

The result of Theorem 1.1, Corollary 1.1, and Theorem 1.2 suggest a limitation of using the inner product similarity structure of HDC encodings when applied to images: the spatial relationship between pixels is lost. In images, the spatial relation between pixels contains

Table 3: 1D-Block Diagonal Shift Laplace-HDC with exact and approximate SVD features

	Exact	Approx $d = 100$	Approx $d = 200$
Float - Fashion MNIST	84.243 ± 0.005	85.553 ± 0.003	85.667 ± 0.003
Binary - Fashion MNIST	84.415 ± 0.011	83.507 ± 0.019	85.703 ± 0.008
Float - MNIST	94.766 ± 0.004	96.610 ± 0.002	95.886 ± 0.001
Binary - MNIST	92.894 ± 0.007	94.017 ± 0.018	94.615 ± 0.003

meaningful information. For example, a group of black pixels may indicate structure, such as a digit, while scattered black pixels may be noise. Note that the ℓ_1-norm is invariant to permutations

$$\|\boldsymbol{x}\|_1 = \sum_{i=1}^{d} |\boldsymbol{x}(i)| = \sum_{i=1}^{d} |\boldsymbol{x}(\sigma(i))| = \|\boldsymbol{x} \circ \sigma\|_1,$$

for any fixed permutation σ of $\{1, \ldots, d\}$. Thus, by Theorem 1.1, the expected similarity structure of the HDC embedding is invariant to a global permutation of the pixels of all the images. One basic way to encode spatial information is to use convolutional features. As a basic demonstration, we consider the 9 Haar wavelet matrices of dimension 4×4; see Figure 2.

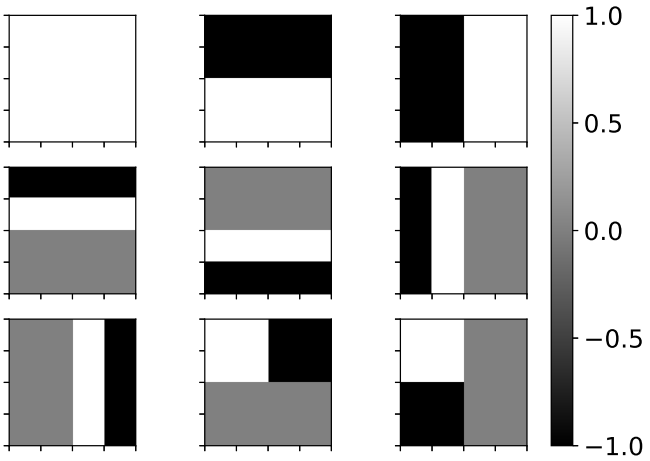

Figure 2: Collection of 9 Haar convolution matrices of dimension 4×4.

Convolving these 9 filters of dimension 4×4 with an $L \times L$ image with stride s creates

$$n = 9((L-4)/s + 1)^2$$

convolutional Haar features, where the stride is the amount of each filter is shifted in each direction when convolving over the data. Each feature coordinate is mapped to the interval $[0, 255]$ by an affine transformation (determined from the training data), which is rounded to an integer in $\{0, 1, \ldots, 255\}$. These integer features are used in the same Laplce-HDC methodology described in §2.2; the results are reported in Table 4.

Table 4: Laplace-HDC with Haar convolutional features

	1D-Cyclic	1D-Block
Float SGD - Fashion MNIST	88.67 ± 0.30	87.86 ± 0.40
Binary SGD - Fashion MNIST	86.65 ± 0.36	85.63 ± 0.56
Float SGD - MNIST	96.40 ± 0.28	96.22 ± 0.29
Binary SGD - MNIST	95.17 ± 0.44	94.85 ± 0.50

More generally, using features from a trained convolutional neural network is possible and would improve the accuracy even further; see §3 for further discussion.

2.5 Comparison to Other Methods

In this section, we implement our proposed framework and compare it to some of the relevant works in the literature, such as RFF-HDC, OnlineHD, and (Extended) HoloGN. In the sequel, we first briefly overview each of these methods and then report their performance on standard datasets such as MNIST and FashionMNIST. To present the results in a reliable format, the experiments are performed 50 times, and mean accuracies along with standard deviation of the accuracies and the histograms are reported in Figure 3 and Table 5.

RFF-HDC This method, presented by Yu et al. (2022) is the most relevant baseline, as it is the basis for our work. While traditional hyperdimensional computing methods (here referred to as Vanilla HDC (Ma et al., 2021) are fast, they suffer from low prediction accuracy. RFF-HDC utilizes similarity matrices in a similar way to Random Fourier Features (RFF) to construct the hypervectors, which helps outperform the state-of-the-art HDC schemes. A pseudo-code that constructs the hypervectors for RFF-HDC was given in §1.1. Note that all the floating point operations are performed during the construction of the hypervectors or learning the models, and ultimately, the models are fully represented and operated in binary mode for inference.

OnlineHD Performing iterative training rather than single-pass training is one approach to boost the accuracy of HDC models, although it increases the time complexity and memory usage, which is costly. Methods such as OnlineHD (Hernández-Cano et al., 2021a) relate the low accuracy of single-pass models to the naive gathering of information from all hypervectors that belong to the same class. This leads to the dominance of the common pattern while downplaying the more uncommon patterns in the data. OnlineHD (Hernández-Cano et al., 2021a) is presented as a single-pass remedy to this problem. Basically, if a hypervector is closely similar to the current state of the class hypervector, then OnlineHD assigns a small weight to it while updating the model in order to decrease its effect, and if the hypervector is distant, the weight increases.

Extended HoloGN Holographic graph neuron (HoloGN) (Kleyko et al., 2017) is an approach designed for character recognition over a dataset of small binary images. HoloGN assigns a randomly generated hypervector to each pixel. Then, a circular shift occurs if the pixel color is white. After this stage, the remaining procedure is similar to vanilla HDC bundling (Binary Majority vote, see §2.1.1). An extension of this approach to operate with

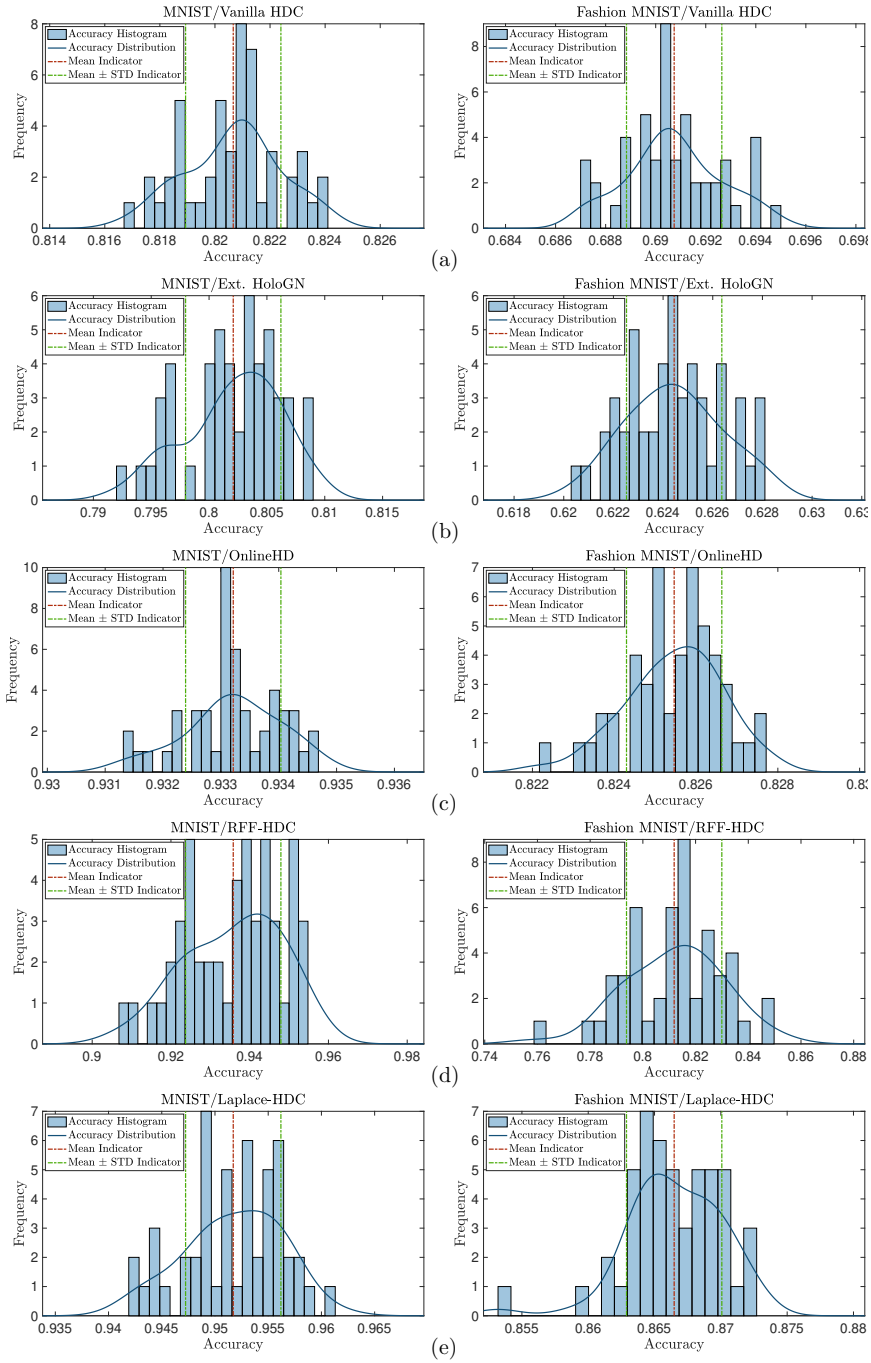

Figure 3: Accuracy histograms of the methods in Table 5 for 50 trials. The mean accuracy and one-standard deviation interval are shown with dashed lines: (a) Vanilla HDC, (b) Extended HoloGN, (c) OnlineHD, (d) RFF-HDC, (e) Laplace-HDC

non-binary datasets such as MNIST was presented in Manabat *et al.* (2019a), which is used as a comparison baseline in our experiments.

The accuracies reported in Table 5 and the histograms depicted in Figure 3 show that Laplace-HDC with convolutional features can outperform the state-of-the-art techniques in terms of the mean accuracy. In terms of the accuracy standard deviation, Laplace-HDC offers a significantly lower deviation compared to the RFF-HDC, with which it shares some foundations.

Table 5: The mean test accuracy (in percentage) of different methods discussed in §2.5 plus and minus one standard deviation, computed for 50 independent experiments. The reported Laplace-HDC uses Haar convolutional features, 1D-Cyclic permutations, and the Binary SGD classifier.

	Vanilla HDC	Ext. HoloGN	OnlineHD	RFF-HDC	Laplace-HDC
MNIST	82.07 ± 0.17	80.21 ± 0.41	93.32 ± 0.08	93.58 ± 1.21	95.17 ± 0.44
Fashion MNIST	69.07 ± 0.19	62.44 ± 0.19	82.55 ± 0.12	81.19 ± 1.81	86.65 ± 0.36

In the next two sections, we explore some other aspects of the proposed binary HDC beyond the accuracy.

2.6 Robustness to Corruptions

A notable characteristic of binary HDC encodings are their robustness to noise: these binary encodings can remain effective even in the presence of corrupted bits. To demonstrate this robustness property for Laplace-HDC, we perform an experiment where the classification task is stress-tested by randomly corrupting a proportion of the bits of the binary encoding.

In this experiment, for each $\boldsymbol{x} \in \mathcal{X}$ the corresponding encoded hypervector $\boldsymbol{\psi}_{\boldsymbol{x}} \in \{-1, 1\}^N$ is corrupted by flipping k of the N bits of the encoded vector to generate a corrupted hypervector $\widetilde{\boldsymbol{\psi}}_{\boldsymbol{x}}$. More precisely, for each $\boldsymbol{x} \in \mathcal{X}$ we choose a set $\{i_1, \ldots, i_k\}$ from $\{1, \ldots, N\}$ independently and uniformly at random without replacement and set

$$\widetilde{\boldsymbol{\psi}}_{\boldsymbol{x}}(i_j) = -\boldsymbol{\psi}_{\boldsymbol{x}}(i_j), \quad \text{for} \quad j = 1, \ldots, k.$$

After this step, the class of $\widetilde{\boldsymbol{\psi}}_{\boldsymbol{x}}$ is inferred using a classifier model which was trained on the uncorrupted data $\boldsymbol{\psi}_{\boldsymbol{x}}$. In this fashion, we are able to determine the degree to which corruption affects classification accuracy. In this experiment, we use the Binary SGD classifier, see §2.1.2. We report results based on the ratio k/N of corrupted bits to the bits in the encoding. When $k/N = 0$, the embedding $\boldsymbol{\psi}_{\boldsymbol{x}}$ is not altered; the method and classification accuracy are the same as reported in §2.2. When $k/N = 1/2$, half the bits are randomly corrupted, and the corrupted embedding $\widetilde{\boldsymbol{\psi}}_{\boldsymbol{x}}$ and original embedding $\boldsymbol{\psi}_{\boldsymbol{x}}$ become uncorrelated. Figure 4 shows the degradation pattern of the Laplace-HDC accuracy as a function of bit error rate. The experiment is performed for the FashionMNIST data for $k = 0, 2, ..., 5000$, and hyperparameters λ and N are set in the same way described in §2.2. Each experiment is performed multiple times, and in addition to the mean accuracy, an uncertainty region of radius three standard deviations is depicted around the mean accuracy plot. One can see that with almost up to a 25% bit error rate (which corresponds to 50% of the worst possible corruption), the classification accuracy confidently maintains a value above 80%.

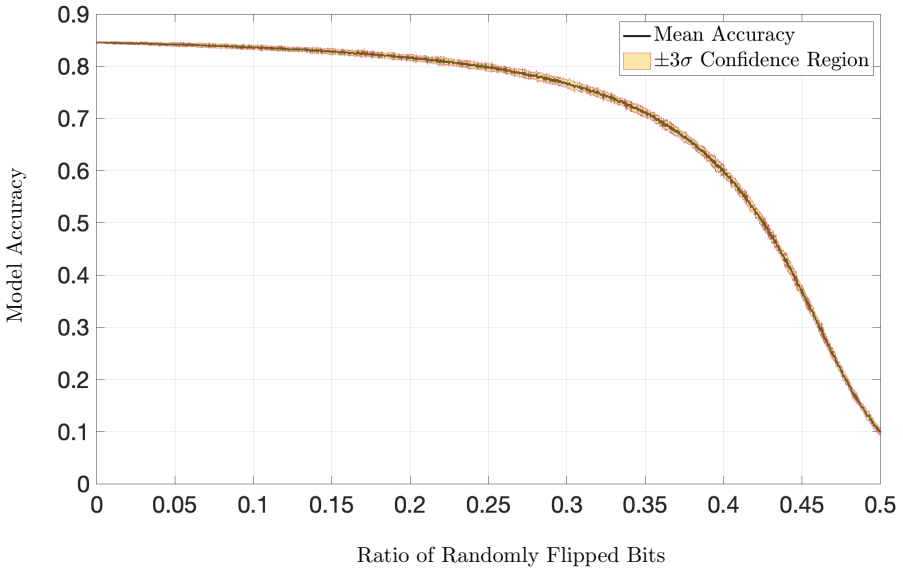

Figure 4: The robustness of the proposed HDC formulation to noise: in each hypervector, a portion of the bits are randomly flipped, and the accuracy is evaluated for the noisy model. The narrow shaded region around the accuracy curve is ±3 times the standard deviation of the accuracy. One can notice that up to almost 25% bit error rate, the accuracy does not drop below 80%.

2.7 Translation Equivariance

In this section, we describe how the 2D-Cyclic family of permutations defined in (18) encodes spatial information and leads to interesting visualizations. In particular, we consider the encoding $\{1,\dots,m\}^{L\times L} \to \{-1,+1\}^{M\times M}$ by

$$\boldsymbol{x} \mapsto \boldsymbol{\psi}_{\boldsymbol{x}} = \bigodot_{i,j=1}^{M} \boldsymbol{T}_{i,j}^{\text{2D-Cyclic}} \boldsymbol{v}_{\boldsymbol{x}(i,j)}, \tag{20}$$

which is translation-equivariant up to boundary effects, see §1.7. Consider examples from the FashionMNIST dataset; see Figure 5.

Let $\boldsymbol{z}$ denote the all zero image $\boldsymbol{z}(i,j) = 0$ for $i,j = 1,\dots,L$, and $\boldsymbol{\psi}_{\boldsymbol{z}}$ be the encoding of the all zero image. To visualize the hypervector encodings of these images, we plot

$$\boldsymbol{\psi}_{\boldsymbol{x}_1} \odot \boldsymbol{\psi}_{\boldsymbol{z}}, \dots, \boldsymbol{\psi}_{\boldsymbol{x}_{10}} \odot \boldsymbol{\psi}_{\boldsymbol{z}},$$

see Figure 6.

Informally speaking, this image consists of translated versions of thresholded versions of the original. The images are overlapping, which captures some autocorrelation of the image with itself when inner products are computed. We note that classification performance seems higher when parameters are tuned so that there is overlap. The number of images in the HDC encoding is controlled by the bandwidth parameter λ. To make another visualization, we can look at the class averages of these images; see Figure 7.

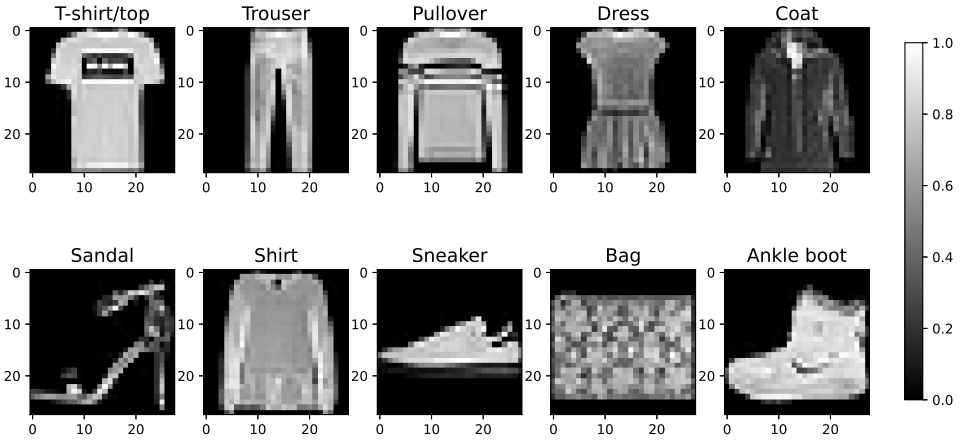

Figure 5: One example from each class in Fashion MNSIT dataset

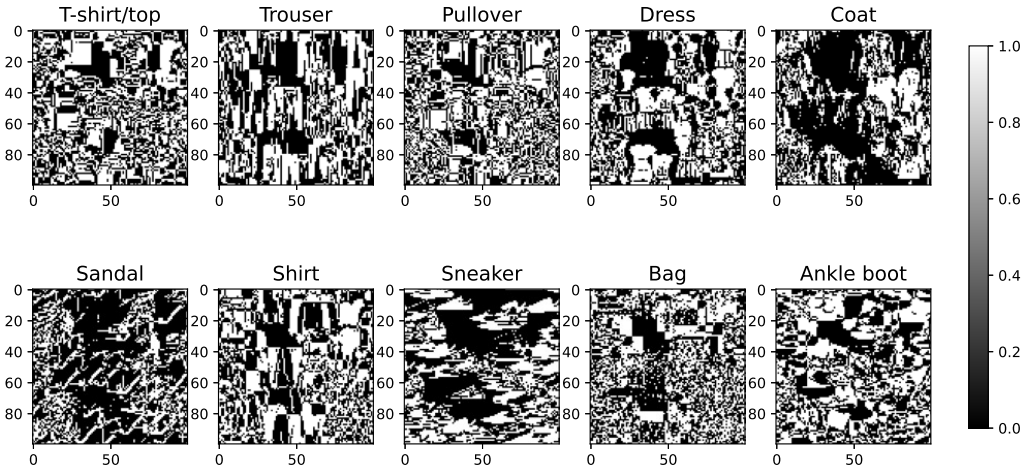

Figure 6: We plot $\boldsymbol{\psi}_{\boldsymbol{x}_j} \odot \boldsymbol{\psi}_{\boldsymbol{z}}$ for the class example images $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{10}$ from Figure 5.

Remark 2.1 (An alternative way to encode spatial information). Recall an implication of our main result Theorem 1.1 is that the binding operation $\boldsymbol{x} \mapsto \boldsymbol{\psi}_{\boldsymbol{x}}$ defined in (1) does not encode spatial information through inner products, see Remark 1.6. One way to address this limitation is to perform feature extraction before the binding operation. For example, in §2.4, we demonstrate that even a simple method of encoding spatial information, such as using Haar convolution matrices, increases the classification accuracy. In this section, we have illustrated an alternative way the binding operation can encode spatial information for

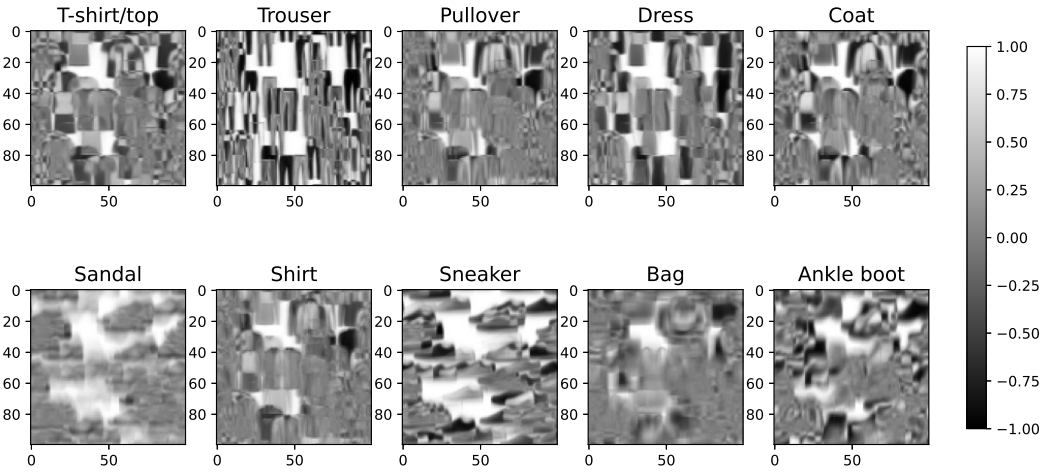

Figure 7: We average $\frac{1}{\#(C_j)} \sum_{\boldsymbol{x} \in C_j} \boldsymbol{\psi}_{\boldsymbol{x}} \odot \boldsymbol{\psi}_{\boldsymbol{z}}$ where C_j is the j-th class, for $j = 1, \ldots, 10$.

appropriate choices of permutation. By using a 2D-Cyclic permutation (20) and taking the entrywise product with the all zero encoding $\boldsymbol{\psi}_{\boldsymbol{z}}$ we illustrate that some spatial information is maintained, see Figures 6 and 7. This spatial information could be extracted by, for example, by using convolution features on the encoding space. Thus, (20) demonstrates another method for improving the classification accuracy of hyperdimensional computing for image data.

Remark 2.2 (Limitations of the high-dimension of HDC encodings). Hyperdimensional computing schemes are inherently high-dimensional as they seek to mimic the brain by using high-dimensional vectors with simple arithmetic operations instead of floating point operations. While the bit-wise operations used in HDC are highly parallelizable, the memory requirements may be challenging for implementations on resource-constrained edge devices. In this paper, since our goal was to understand the geometry of the HDC encoding space, we considered a fixed hyperdimension of $N \approx 10^4$ for all numerical experiments. However, it may be possible to post-process the encoding to reduce this dimension further or incorporate our theoretical approach in a scheme designed to reduce the dimension, such as the Low-Dimensional Computing (LDC) method of (Duan, Xu, & Ren, 2022b).

3. Discussion

This paper introduces Laplace-HDC, a binary HDC scheme motivated by the geometry of hypervector data encodings. We build upon the work of Yu *et al.* (2022), by considering the inner product structure of hypervector encodings resulting from the binding operation for hypervectors with a covariance structure.

We show that the Laplace kernel (rather than the Gaussian kernel) is a natural choice for the covariance structure of the hypervectors used in this construction. In this case, we show that the inner product of the hypervector encodings of data is related to an ℓ_1-norm Laplace kernel between data points, which motivates a method for setting the bandwidth parameter $\lambda > 0$ in this construction. These observations lead to a practical binary HDC scheme, which we call Laplace-HDC.

Our results also indicate a limitation of binary HDC schemes of this type for image data: the spatial relationship between pixels is lost. More precisely, our results show that the inner product structure of hypervector encodings is invariant to global permutations of the data. We demonstrate that when spatial relationships are encoded, even in an elementary way, such as through convolutional Haar features (which are not invariant to global permutations of the pixels), the accuracy of binary HDC improves for image data.

We note that more complicated feature extraction methods could be used to increase the performance further. For example, the features derived from the output of one or more layers of a convolutional neural network trained on image data could be used. The fact that binary HDC schemes of this type are invariant to global permutations of the pixels can be viewed both as a limitation or a feature of binding-based binary HDC encoding schemes.

We emphasize that our theoretical results only say that spatial relationships are not encoded in the inner product structure of the binding operation. It may be possible to recover spatial information via another method. We illustrate such a method when we define a translation-equivariant binary HDC encoding scheme for images, which is a potential direction for future work.

We note that the Trace-Orthogonal assumption that we make on the families of permutation matrices we consider may be overly restrictive. We performed some limited experiments using families of permutations, which are each sampled independently and uniformly, which achieved similar accuracy to the Trace-Orthogonal families of permutation we considered (1D-Cyclic, 1D-Block, 2D-Cyclic, 2D-Block). However, it should be noted that each of the families of permutations we considered has efficient implementations that maintain memory locality when encoding batched images. In contrast, performing a uniformly random permutation is orders of magnitude slower due to a lack of memory locality. However, there may be pseudo-random permutations that can be efficiently implemented that are interesting to consider.

Future work on Laplace-HDC could investigate methods to refine class representative selection and reduce model hyperdimensionality. LeHDC, an HDC method developed by Duan et al., (2022a), describes a principled method to refine hypervector selection in an existing HDC model, contrasting with existing class hypervector selection methods which are often heuristic in nature. This method could be applied to our method to develop a neural network version of Laplace-HDC. Additionally, another recent paper by Duan et al. (2022b) develops a neural-network-based approach to generating hypervectors which results in a model requiring significantly fewer hyperdimensions while maintaining high performance on classification tasks. This approach could offer motivation for a neural-network-based approach to Laplace-HDC.

Acknowledgments

The authors thank Peter Cowal for useful discussions about the paper. Authors Saeid Pourmand and Wyatt Whiting contributed equally to this project.

Appendix A. Proof of Analytic Results

A.1 Proof of Theorem 1.1

Proof of Theorem 1.1. By the definition (1) of the map $\boldsymbol{x} \mapsto \boldsymbol{\psi}_{\boldsymbol{x}}$ we have

$$\begin{aligned}
S(\boldsymbol{x},\boldsymbol{y}) &= \frac{1}{N}\mathbb{E}\,\mathrm{Tr}\left(\boldsymbol{\psi}_{\boldsymbol{x}}\boldsymbol{\psi}_{\boldsymbol{y}}^{\top}\right) \\
&= \frac{1}{N}\mathbb{E}\,\mathrm{Tr}\left(\left(\bigodot_{i=1}^{d}\boldsymbol{\Pi}_i\boldsymbol{v}_{\boldsymbol{x}(i)}\right)\left(\bigodot_{i=1}^{d}\boldsymbol{v}_{\boldsymbol{y}(i)}^{\top}\boldsymbol{\Pi}_i^{\top}\right)\right) \\
&= \frac{1}{N}\mathbb{E}\,\mathrm{Tr}\left(\bigodot_{i=1}^{d}\boldsymbol{\Pi}_i\boldsymbol{v}_{\boldsymbol{x}(i)}\boldsymbol{v}_{\boldsymbol{y}(i)}^{\top}\boldsymbol{\Pi}_i^{\top}\right) \\
&= \frac{1}{N}\sum_{j=1}^{N}\mathbb{E}\left(\prod_{i=1}^{d}\boldsymbol{e}_j^{\top}\boldsymbol{\Pi}_i\boldsymbol{v}_{\boldsymbol{x}(i)}\boldsymbol{v}_{\boldsymbol{y}(i)}^{\top}\boldsymbol{\Pi}_i^{\top}\boldsymbol{e}_j\right) \\
&= \frac{1}{N}\sum_{j=1}^{N}\prod_{i=1}^{d}\mathbb{E}\left(\boldsymbol{e}_j^{\top}\boldsymbol{\Pi}_i\boldsymbol{v}_{\boldsymbol{x}(i)}\boldsymbol{v}_{\boldsymbol{y}(i)}^{\top}\boldsymbol{\Pi}_i^{\top}\boldsymbol{e}_j\right), \\
&= \frac{1}{N}\sum_{j=1}^{N}\prod_{i=1}^{d}\boldsymbol{K}(\boldsymbol{x}(i),\boldsymbol{y}(i)) \\
&= \prod_{i=1}^{d}\boldsymbol{K}(\boldsymbol{x}(i),\boldsymbol{y}(i)). \qquad (21)
\end{aligned}$$

In the chain of equalities above, the third equality holds since for arbitrary vectors $\boldsymbol{u}_i, \boldsymbol{v}_i, \mathbb{R}^N$,

$$\bigodot_{i=1}^{d}\boldsymbol{u}_i\bigodot_{i'=1}^{d}\boldsymbol{v}_{i'}^{\top} = \bigodot_{i=1}^{d}\boldsymbol{u}_i\boldsymbol{v}_i^{\top},$$

where the right-side $\bigodot$ represents a straightforward generalization of Hadamard product from vectors to matrices. Moreover, by the construction of $\boldsymbol{v}_k$, the elements $\boldsymbol{v}_k(j)$ and $\boldsymbol{v}_{k'}(j')$ are statistically independent whenever $j \neq j'$. As a result, since the permutation matrices $\boldsymbol{\Pi}_i$ are non-overlapping, the factors $\boldsymbol{e}_j^{\top}\boldsymbol{\Pi}_i\boldsymbol{v}_{\boldsymbol{x}(i)}\boldsymbol{v}_{\boldsymbol{y}(i)}^{\top}\boldsymbol{\Pi}_i^{\top}\boldsymbol{e}_j$ and $\boldsymbol{e}_j^{\top}\boldsymbol{\Pi}_{i'}\boldsymbol{v}_{\boldsymbol{x}(i')}\boldsymbol{v}_{\boldsymbol{y}(i')}^{\top}\boldsymbol{\Pi}_{i'}^{\top}\boldsymbol{e}_j$ become independent whenever $i \neq i'$, which justifies the fifth equality. Finally, the sixth equality is a straightforward implication of (6).

Next, we bound the variance. First, we compute the second moment

$$\begin{aligned}\mathbb{E}\left(\frac{\boldsymbol{\psi}_{\boldsymbol{x}}^\top \boldsymbol{\psi}_{\boldsymbol{y}}}{N}\right)^2 &= \frac{1}{N^2}\mathbb{E}\left(\sum_{j=1}^{N}\prod_{i=1}^{d} \boldsymbol{e}_j^\top \boldsymbol{\Pi}_i \boldsymbol{v}_{\boldsymbol{x}(i)} \boldsymbol{v}_{\boldsymbol{y}(i)}^\top \boldsymbol{\Pi}_i^\top \boldsymbol{e}_j\right)^2 \\ &= \frac{1}{N^2}\mathbb{E}\left(\sum_{j=1}^{N}\prod_{i=1}^{d} \boldsymbol{e}_j^\top \boldsymbol{\Pi}_i \boldsymbol{v}_{\boldsymbol{x}(i)} \boldsymbol{v}_{\boldsymbol{y}(i)}^\top \boldsymbol{\Pi}_i^\top \boldsymbol{e}_j\right)\left(\sum_{j'=1}^{N}\prod_{i'=1}^{d} \boldsymbol{e}_{j'}^\top \boldsymbol{\Pi}_{i'} \boldsymbol{v}_{\boldsymbol{x}(i')} \boldsymbol{v}_{\boldsymbol{y}(i')}^\top \boldsymbol{\Pi}_{i'}^\top \boldsymbol{e}_{j'}\right) \\ &= \frac{1}{N^2}\sum_{j=1}^{N}\sum_{j'=1}^{N}\mathbb{E}\left(\prod_{i=1}^{d} \boldsymbol{e}_j^\top \boldsymbol{\Pi}_i \boldsymbol{v}_{\boldsymbol{x}(i)} \boldsymbol{v}_{\boldsymbol{y}(i)}^\top \boldsymbol{\Pi}_i^\top \boldsymbol{e}_j \prod_{i'=1}^{d} \boldsymbol{e}_{j'}^\top \boldsymbol{\Pi}_{i'} \boldsymbol{v}_{\boldsymbol{x}(i')} \boldsymbol{v}_{\boldsymbol{y}(i')}^\top \boldsymbol{\Pi}_{i'}^\top \boldsymbol{e}_{j'}\right).\end{aligned} \tag{22}$$

Define the $N \times N$ matrix

$$\boldsymbol{Q} := \sum_{i'=1}^{d}\sum_{i=1}^{d} \boldsymbol{\Pi}_i \boldsymbol{\Pi}_{i'}^\top,$$

and accordingly, define

$$\Omega := \left\{(j, j') : \boldsymbol{Q}(j, j') > 0\right\},$$

where $\boldsymbol{Q}(j, j')$ is the (j, j')-th element of $\mathbf{Q}$. One can split the summation above over Ω and Ω^c as:

$$\mathbb{E}\left(\frac{\boldsymbol{\psi}_{\boldsymbol{x}}^\top \boldsymbol{\psi}_{\boldsymbol{y}}}{N}\right)^2 = E_\Omega + E_{\Omega^c}, \tag{23}$$

where for $\mathcal{X} = \Omega, \Omega^c$:

$$E_{\mathcal{X}} = \frac{1}{N^2}\mathbb{E}\sum_{(j,j')\in\mathcal{X}}\prod_{i=1}^{d} \boldsymbol{e}_j^\top \boldsymbol{\Pi}_i \boldsymbol{v}_{\boldsymbol{x}(i)} \boldsymbol{v}_{\boldsymbol{y}(i)}^\top \boldsymbol{\Pi}_i^\top \boldsymbol{e}_j \prod_{i'=1}^{d} \boldsymbol{e}_{j'}^\top \boldsymbol{\Pi}_{i'} \boldsymbol{v}_{\boldsymbol{x}(i')} \boldsymbol{v}_{\boldsymbol{y}(i')}^\top \boldsymbol{\Pi}_{i'}^\top \boldsymbol{e}_{j'}.$$

The summand is clearly upper-bounded by 1, so the summation of terms over Ω is bounded by

$$E_\Omega \le \frac{\sum_{(j,j')\in\Omega} 1}{N^2} = \frac{|\Omega|}{N^2} = \frac{\gamma_{\mathcal{P}}}{N^2}, \tag{24}$$

where the final equality follows from the definition of $\gamma_{\mathcal{P}}$. On the other hand, for $(j, j') \in \Omega^c$ we have

$$\mathbb{E}\prod_{i=1}^{d} \boldsymbol{e}_j^\top \boldsymbol{\Pi}_i \boldsymbol{v}_{\boldsymbol{x}(i)} \boldsymbol{v}_{\boldsymbol{y}(i)}^\top \boldsymbol{\Pi}_i^\top \boldsymbol{e}_j \prod_{i'=1}^{d} \boldsymbol{e}_{j'}^\top \boldsymbol{\Pi}_{i'} \boldsymbol{v}_{\boldsymbol{x}(i')} \boldsymbol{v}_{\boldsymbol{y}(i')}^\top \boldsymbol{\Pi}_{i'}^\top \boldsymbol{e}_{j'} = S(\boldsymbol{x}, \boldsymbol{y})^2. \tag{25}$$

To see why (25) holds, we start by showing that if $(j, j') \in \Omega^c$, then

$$\boldsymbol{e}_j^\top \boldsymbol{\Pi}_i \neq \boldsymbol{e}_{j'}^\top \boldsymbol{\Pi}_{i'}, \quad \forall i, i' \in \{1, \ldots, d\}. \tag{26}$$

Suppose not, that is, suppose $\boldsymbol{e}_j^\top \boldsymbol{\Pi}_{i_0} = \boldsymbol{e}_{j'}^\top \boldsymbol{\Pi}_{i_0'}$, for some $i_0, i_0' \in \{1, \ldots, d\}$. Then,

$$0 < \boldsymbol{e}_j^\top \boldsymbol{\Pi}_{i_0} \boldsymbol{\Pi}_{i_0'}^\top \boldsymbol{e}_{j'} \le \boldsymbol{e}_j^\top \sum_i \sum_{i'} \boldsymbol{\Pi}_i \boldsymbol{\Pi}_{i'}^\top \boldsymbol{e}_{j'} = \boldsymbol{Q}(j, j'),$$

which contradicts the fact that $(j, j') \in \Omega^c$. Recall that, by construction, the elements $\boldsymbol{v}_k(j)$ and $\boldsymbol{v}_{k'}(j')$ are statistically independent whenever $j \neq j'$. By (26), it follows that $\boldsymbol{e}_j^\top \boldsymbol{\Pi}_i \boldsymbol{v}_{\boldsymbol{x}(i)} \boldsymbol{v}_{\boldsymbol{y}(i)}^\top \boldsymbol{\Pi}_i^\top \boldsymbol{e}_j$ and $\boldsymbol{e}_{j'}^\top \boldsymbol{\Pi}_{i'} \boldsymbol{v}_{\boldsymbol{x}(i')} \boldsymbol{v}_{\boldsymbol{y}(i')}^\top \boldsymbol{\Pi}_{i'}^\top \boldsymbol{e}_{j'}$ are independent, for all $i, i' \in \{1, \dots, d\}$. Thus, we can take the expectation of the product of i and i' separately and use the fact that

$$\prod_{i=1}^{d} \mathbb{E}\left(\boldsymbol{e}_j^\top \boldsymbol{\Pi}_i \boldsymbol{v}_{\boldsymbol{x}(i)} \boldsymbol{v}_{\boldsymbol{y}(i)}^\top \boldsymbol{\Pi}_i^\top \boldsymbol{e}_j\right) = S(\mathbf{x}, \mathbf{y}),$$

to deduce (25). Combining (23), (24), and (25) gives

$$\mathbb{E}\left(\frac{\boldsymbol{\psi}_{\boldsymbol{x}}^\top \boldsymbol{\psi}_{\boldsymbol{y}}}{N}\right)^2 \leq \frac{\gamma_{\mathcal{P}}}{N^2} + \frac{1}{N^2} \sum_{(j,j') \in \Omega^c} S(\boldsymbol{x}, \boldsymbol{y})^2 = \frac{\gamma_{\mathcal{P}}}{N^2} + \frac{N^2 - \gamma_{\mathcal{P}}}{N^2} S(\boldsymbol{x}, \boldsymbol{y})^2.$$

Finally, we have

$$\begin{aligned} \operatorname{Var}\left(\frac{\boldsymbol{\psi}_{\boldsymbol{x}}^\top \boldsymbol{\psi}_{\boldsymbol{y}}}{N}\right) &= \mathbb{E}\left(\frac{\boldsymbol{\psi}_{\boldsymbol{x}}^\top \boldsymbol{\psi}_{\boldsymbol{y}}}{N}\right)^2 - \left(\mathbb{E}\frac{\boldsymbol{\psi}_{\boldsymbol{x}}^\top \boldsymbol{\psi}_{\boldsymbol{y}}}{N}\right)^2 \\ &\leq \frac{\gamma_{\mathcal{P}}}{N^2} + \frac{N^2 - \gamma_{\mathcal{P}}}{N^2} S(\boldsymbol{x}, \boldsymbol{y})^2 - S(\boldsymbol{x}, \boldsymbol{y})^2 \\ &= \frac{\gamma_{\mathcal{P}}}{N^2}(1 - S(\boldsymbol{x}, \boldsymbol{y})^2) \\ &\leq \frac{2\gamma_{\mathcal{P}}}{N^2}(1 - S(\boldsymbol{x}, \boldsymbol{y})), \end{aligned} \tag{27}$$

where the final inequality follows from using the fact that $1 + S(\boldsymbol{x}, \boldsymbol{y}) \leq 2$. This completes the proof.

□

Remark A.1 (Sharpness of proof of Theorem 1.1). From (22) we have

$$\mathbb{E}\left(\frac{\boldsymbol{\psi}_{\boldsymbol{x}}^\top \boldsymbol{\psi}_{\boldsymbol{y}}}{N}\right)^2 = \frac{1}{N^2} \sum_{j=1}^{N} \sum_{j'=1}^{N} \mathbb{E}\left(\prod_{i=1}^{d} X_{j,i} \prod_{i'=1}^{d} X_{j',i'}\right),$$

where

$$X_{j,i} = \boldsymbol{e}_j^\top \boldsymbol{\Pi}_i \boldsymbol{v}_{\boldsymbol{x}(i)} \boldsymbol{v}_{\boldsymbol{y}(i)}^\top \boldsymbol{\Pi}_i^\top \boldsymbol{e}_j.$$

In the following, we show this estimate can be refined leading to a more complicated statement. Let

$$\Omega_{j,j'} = \left\{ i \in \{1, \dots, d\} : \boldsymbol{e}_j^\top \boldsymbol{\Pi}_i \sum_{i'=1}^{d} \boldsymbol{\Pi}_{i'}^\top \boldsymbol{e}_{j'} > 0 \right\}.$$

Using this notation, we have

$$\mathbb{E}\left(\prod_{i=1}^{d} X_{j,i} \prod_{i'=1}^{d} X_{j',i'}\right) = \prod_{i \in \Omega_{j,j'}^c} \mathbb{E}\left(X_{j,i}\right) \prod_{i' \in \Omega_{j',j}^c} \mathbb{E}\left(X_{j',i'}\right) \mathbb{E}\left(\prod_{i \in \Omega_{j,j'}} X_{j,i} \prod_{i' \in \Omega_{j',j}} X_{j',i'}\right), \tag{28}$$

where we used the fact that $X_{j,i}$ is independent of all other random variables in the product when $i \in \Omega^c_{j,j'}$, and likewise, by symmetry, $X_{j,i'}$ is independent of all other random variables in the product when $i' \in \Omega_{j',j}$. We have

$$\prod_{i\in\Omega^c_{j,j'}} \mathbb{E}X_{j,i} \prod_{i'\in\Omega^c_{j',j}} \mathbb{E}X_{j',i'} = \prod_{i\in\Omega^c_{j,j'}} \boldsymbol{K}(\boldsymbol{x}(i),\boldsymbol{y}(i)) \prod_{i'\in\Omega^c_{j',j}} \boldsymbol{K}(\boldsymbol{x}(i'),\boldsymbol{y}(i')),$$

while the third term on the right-hand side of (28) is bounded by 1. In fact, this bound is sharp in certain cases, for example, in the case when $\boldsymbol{x}$ and $\boldsymbol{y}$ are constant. In summary, we have

$$\mathbb{E}\left(\frac{\boldsymbol{\psi}_{\boldsymbol{x}}^\top \boldsymbol{\psi}_{\boldsymbol{y}}}{N}\right)^2 \leq \frac{1}{N^2}\sum_{j=1}^N\sum_{j'=1}^N \prod_{i\in\Omega^c_{j,j'}} \boldsymbol{K}(\boldsymbol{x}(i),\boldsymbol{y}(i)) \prod_{i'\in\Omega^c_{j',j}} \boldsymbol{K}(\boldsymbol{x}(i'),\boldsymbol{y}(i')).$$

Note that with this notation $\Omega_{j,j'}$ is the set of "bad permutations", which cause terms in the product of random variables to be dependent. When $\Omega_{j,j'} = \emptyset$ and $\Omega_{j',j} = \emptyset$ we have

$$\prod_{i\in\Omega^c_{j,j'}} \boldsymbol{K}(\boldsymbol{x}(i),\boldsymbol{y}(i)) \prod_{i'\in\Omega^c_{j',j}} \boldsymbol{K}(\boldsymbol{x}(i'),\boldsymbol{y}(i')) = S(\boldsymbol{x},\boldsymbol{y})^2.$$

Alternatively, if there is only one bad permutation in each set $\Omega_{j,j'} = \{i_0\}$ and $\Omega_{j',j} = \{i_0'\}$ we have

$$\prod_{i\in\Omega^c_{j,j'}} \boldsymbol{K}(\boldsymbol{x}(i),\boldsymbol{y}(i)) \prod_{i'\in\Omega^c_{j',j}} \boldsymbol{K}(\boldsymbol{x}(i'),\boldsymbol{y}(i')) = \frac{S(\boldsymbol{x},\boldsymbol{y})^2}{\boldsymbol{K}(\boldsymbol{x}(i_0),\boldsymbol{y}(i_0))\boldsymbol{K}(\boldsymbol{x}(i_0'),\boldsymbol{y}(i_0'))}.$$

Previously, we performed a global analysis, which only looked at the number of j, j' for which there are any bad permutations (see Ω below). This refined analysis shows that the number of bad permutations also matters. The block-based permutation schemes minimize the number of j, j' for which $\Omega_{j,j'} > 0$, but do this bad trying to maximize $|\Omega_{j,j'}|$ whenever $\Omega_{j,j'} > 0$.

A.2 Proof of Theorem 1.2

Proof of Theorem 1.2. If $\boldsymbol{K}_\alpha$ is defined by (14), then

$$\boldsymbol{W}_\alpha(i,j) = \exp\left(-\frac{\pi^2}{8}\lambda|i-j|^{2\alpha}\right),$$

which is positive semi-definite by Lemma 1.2 and the fact that $\alpha \in (0,1]$. Note that

$$\frac{2}{\pi}\arcsin\left(\exp\left(-x^2\right)\right) = 1 - \frac{2\sqrt{2}x}{\pi} + \mathcal{O}(x^3),$$

as $x \to 0$. If $\lambda|i-j|^\alpha \leq \varepsilon$, then it follows that the kernel $\boldsymbol{K}_\alpha$ defined in (14) satisfies

$$\boldsymbol{K}_\alpha(i,j) = 1 - \lambda|i-j|^\alpha + \mathcal{O}(\varepsilon^3). \tag{29}$$

Write

$$S_\alpha(\boldsymbol{x}, \boldsymbol{y}) = \prod_{i=1}^{d} \boldsymbol{K}_\alpha(\boldsymbol{x}(i), \boldsymbol{y}(j)) = \exp\left(\sum_{i=1}^{d} \ln\left(\boldsymbol{K}_\alpha(\boldsymbol{x}(i), \boldsymbol{y}(j))\right)\right).$$

Using the series expansions

$$\ln(1-x) = -x + \mathcal{O}(x^2), \quad \text{and} \quad \exp(x) = 1 + \mathcal{O}(x),$$

as $x \to 0$, together with (29) gives

$$\exp\left(\sum_{i=1}^{d} \ln\left(\boldsymbol{K}_\alpha(\boldsymbol{x}(i), \boldsymbol{y}(j))\right)\right) = \exp\left(-\lambda \sum_{i=1}^{d} |\boldsymbol{x}(i) - \boldsymbol{y}(i)|^\alpha\right)\left(1 + \mathcal{O}(\varepsilon^2 d)\right).$$

That is,

$$S_\alpha(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\lambda \|\boldsymbol{x} - \boldsymbol{y}\|_\alpha^\alpha\right)\left(1 + \mathcal{O}(\varepsilon^2 d)\right)$$

which completes the proof.

□

References

Basaklar, T., Tuncel, Y., Narayana, S. Y., Gumussoy, S., & Ogras, U. Y. (2021). Hypervector design for efficient hyperdimensional computing on edge devices. In *tinyML 2021 Research Symposium*.

Byerly, A., Kalganova, T., & Dear, I. (2021). No routing needed between capsules. *Neurocomputing*, *463*, 545–553.

Cao, L. (2006). Singular value decomposition applied to digital image processing. *Division of Computing Studies, Arizona State University Polytechnic Campus, Mesa, Arizona State University polytechnic Campus*, pp. 1–15.

Chen, L., & Xu, S. (2020). Deep neural tangent kernel and Laplace kernel have the same RKHS. *arXiv e-print arXiv:2009.10683*.

Chen, Z. (2018). Singular value decomposition and its applications in image processing. In *Proceedings of the 2018 1st International Conference on Mathematics and Statistics*, pp. 16–22.

Chuang, Y.-C., Chang, C.-Y., & Wu, A.-Y. A. (2020). Dynamic hyperdimensional computing for improving accuracy-energy efficiency trade-offs. In *2020 IEEE Workshop on Signal Processing Systems (SiPS)*, pp. 1–5. IEEE.

Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., & Bengio, Y. (2016). Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or −1. *arXiv e-print arXiv:1602.02830*.

Duan, S., Liu, Y., Ren, S., & Xu, X. (2022a). Lehdc: learning-based hyperdimensional computing classifier. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pp. 1111–1116, New York, NY, USA. ACM.

Duan, S., Xu, X., & Ren, S. (2022b). A brain-inspired low-dimensional computing classifier for inference on tiny devices. *arXiv preprint arXiv:2203.04894*.

Ge, L., & Parhi, K. K. (2020). Classification using hyperdimensional computing: A review. *IEEE Circuits and Systems Magazine*, *20*(2), 30–47.

Geifman, A., Yadav, A., Kasten, Y., Galun, M., Jacobs, D., & Ronen, B. (2020). On the similarity between the Laplace and neural tangent kernels. *Advances in Neural Information Processing Systems*, *33*, 1451–1461.

Gupta, S., Morris, J., Imani, M., Ramkumar, R., Yu, J., Tiwari, A., Aksanli, B., & Rosing, T. Š. (2020). Thrifty: Training with hyperdimensional computing across flash hierarchy. In *Proceedings of the 39th International Conference on Computer-Aided Design*, pp. 1–9.

Halko, N., Martinsson, P.-G., & Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, *53*(2), 217–288.

Hassan, E., Halawani, Y., Mohammad, B., & Saleh, H. (2021). Hyper-dimensional computing challenges and opportunities for AI applications. *IEEE Access*, *10*, 97651–97664.

Hernández-Cano, A., Matsumoto, N., Ping, E., & Imani, M. (2021a). OnlineHD: Robust, efficient, and single-pass online learning using hyperdimensional system. In *2021 Design, Automation, and Test in Europe Conference Exhibition (DATE)*, pp. 56–61. IEEE.

Hernández-Cano, A., Matsumoto, N., Ping, E., & Imani, M. (2021b). OnlineHD: Robust, efficient, and single-pass online learning using hyperdimensional system. In *2021 Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pp. 56–61. IEEE.

Hirata, D., & Takahashi, N. (2023). Ensemble learning in cnn augmented with fully connected subnetworks. *IEICE Transactions on Information and Systems*, *106*(7), 1258–1261.

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

Imani, M., Morris, J., Messerly, J., Shu, H., Deng, Y., & Rosing, T. (2019). BRIC: Locality-based encoding for energy-efficient brain-inspired hyperdimensional computing. In *Proceedings of the 56th Annual Design Automation Conference 2019*, pp. 1–6.

Kabir, H. M. D., Abdar, M., Khosravi, A., Jalali, S. M. J., Atiya, A. F., Nahavandi, S., & Srinivasan, D. (2023). SpinalNet: Deep neural network with gradual input. *IEEE transactions on artificial intelligence*, *4*(5), 1165–1177.

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

Karunaratne, G., Le Gallo, M., Cherubini, G., Benini, L., Rahimi, A., & Sebastian, A. (2020). In-memory hyperdimensional computing. *Nature Electronics*, *3*(6), 327–337.

Karunaratne, G., Rahimi, A., Gallo, M. L., Cherubini, G., & Sebastian, A. (2021). Real-time language recognition using hyperdimensional computing on phase-change memory array. In *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pp. 1–1.

Khaleghi, B., Xu, H., Morris, J., & Rosing, T. v. (2021). tiny-HD: Ultra-efficient hyperdimensional computing engine for IoT applications. In *2021 Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pp. 408–413.

Kim, M., & Smaragdis, P. (2016). Bitwise neural networks. *arXiv e-print arXiv:1601.06071*.

Kim, S.-J., Magnani, A., & Boyd, S. (2006). Optimal kernel selection in kernel fisher discriminant analysis. In *Proceedings of the 23rd international conference on Machine learning*, pp. 465–472.

Kim, Y., Imani, M., Moshiri, N., & Rosing, T. (2020). GenieHD: Efficient DNA pattern matching accelerator using hyperdimensional computing. In *2020 Design, Automation and Test in Europe Conference Exhibition (DATE)*, pp. 115–120. IEEE.

Kingma, D. P., & Ba, J. (2017). Adam: A method for stochastic optimization. *arXiv e-print arXiv:1412.6980*.

Kleyko, D., Osipov, E., Senior, A., Khan, A. I., & Sekercioglu, Y. A. (2017). Holographic graph neuron: A bioinspired architecture for pattern processing. *IEEE transaction on neural networks and learning systems*, *28*(6), 1250–1262.

Kleyko, D., Rachkovskij, D., Osipov, E., & Rahimi, A. (2023a). A survey on hyperdimensional computing aka vector symbolic architectures, part ii: Applications, cognitive models, and challenges. *ACM computing surveys*, *55*(9), 1–52.

Kleyko, D., Rachkovskij, D. A., Osipov, E., & Rahimi, A. (2023b). A survey on hyperdimensional computing aka vector symbolic architectures, part i: Models and data transformations. *ACM computing surveys*, *55*(6), 1–40.

Ma, D., Guo, J., Jiang, Y., & Jiao, X. (2021). Hdtest: Differential fuzz testing of brain-inspired hyperdimensional computing. In *Proceedings of the 58th ACM/EDAC/IEEE Design Automation Conference (DAC)*.

Manabat, A. X., Marcelo, C. R., Quinquito, A. L., & Alvarez, A. (2019a). Performance analysis of hyperdimensional computing for character recognition. In *2019 International Symposium on Multimedia and Communication Technology (ISMAC)*, pp. 1–5.

Manabat, A. X., Marcelo, C. R., Quinquito, A. L., & Alvarez, A. (2019b). Performance analysis of hyperdimensional computing for character recognition. In *2019 International Symposium on Multimedia and Communication Technology (ISMAC)*, pp. 1–5. IEEE.

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

Neubert, P., & Schubert, S. (2021). Hyperdimensional computing as a framework for systematic aggregation of image descriptors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16938–16947.

Neubert, P., Schubert, S., & Protzel, P. (2019). An introduction to hyperdimensional computing for robotics. *KI-Künstliche Intelligenz*, *33*(4), 319–330.

Rudelson, M., & Vershynin, R. (2013). Hanson-wright inequality and sub-gaussian concentration. *Electronic communications in probability*, *18*.

Schlegel, K., Neubert, P., & Protzel, P. (2022). A comparison of vector symbolic architectures. *Artificial Intelligence Review*, *55*(6), 4523–4555.

Schoenberg, I. J. (1938). Metric spaces and positive definite functions. *Transactions of the American Mathematical Society*, *44*(3), 522–536.

Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge university press.

Thomas, A., Dasgupta, S., & Rosing, T. (2021). A theoretical perspective on hyperdimensional computing. *Journal of Artificial Intelligence Research*, *72*, 215–249.

Vershynin, R. (2018). *High-dimensional probability: An introduction with applications in data science*, Vol. 47. Cambridge university press.

Yu, T., Zhang, Y., Zhang, Z., & De Sa, C. M. (2022). Understanding hyperdimensional computing for parallel single-pass learning. *Advances in Neural Information Processing Systems*, *35*, 1157–1169.