HDQMF: Holographic Feature Decomposition Using Quantum Algorithms

Prathyush Poduval
University of California, Irvine
Irvine, CA 92617
ppoduval@uci.edu

Zhuowen Zou
University of California, Irvine
Irvine, CA 92617
zhuowez1@uci.edu

Mohsen Imani
University of California, Irvine
Irvine, CA 92617
m.imani@uci.edu

Abstract

This paper addresses the decomposition of holographic feature vectors in Hyperdimensional Computing (HDC) aka Vector Symbolic Architectures (VSA). HDC uses high-dimensional vectors with brain-like properties to represent symbolic information, and leverages efficient operators to construct and manipulate complexly structured data in a cognitive fashion. Existing models face challenges in decomposing these structures, a process crucial for understanding and interpreting a composite hypervector. We address this challenge by proposing the HDC Memorized-Factorization Problem that captures the common patterns of construction in HDC models. To solve this problem efficiently, we introduce HDQMF*, a HyperDimensional Quantum Memorized-Factorization algorithm.* HDQMF *is unique in its approach, utilizing quantum computing to offer efficient solutions. It modifies crucial steps in Grover's algorithm to achieve hypervector decomposition, achieving quadratic speed-up.*

1. Introduction

Hyperdimensional Computing (HDC), synonymously Vector Symbolic Architecture (VSA), has gained much popularity as a framework that provides a natural implementation of cognitive data structure [17]. It represents information using hypervectors, a type of holographic and high-dimensional vector that satisfies brain-inspired properties [14]. It leverages simple and efficient operations - bundle, bind, and permute - that support the three patterns of combinations that cognitive scientists deemed essential in a cognitive data structure: variable binding, sequential structures, and hierarchy [3, 5]. This allows HDC models to represent and manipulate complex and hierarchically structured data interpretably, where the similarity between the hypervectors reflects their structural similarity and componential similarity in a well-defined manner, providing a direct foundation for symbolic logical reasoning.

Recent work has shown great advantages of HDC in enhancing the cognitive capability of neural networks. [10] proposes a neural-vector-symbolic architecture that tackles the binding problem in neural networks by training an encoder network to generate HDC-like representation for subsequence symbolic logic processing, significantly outperforming state-of-the-art pure DNN and neuro-symbolic AI on Raven's progressive matrices task in both accuracy and efficiency scaling. Pioneering the development of a neural-vector-symbolic architecture that addresses the limitations of both neural networks and symbolic methods in accomplishing cognitive tasks, [10] has pointed out many limitations of the current model, one of which is the decomposition problem that recurs in the general usage of HDC.

This paper focuses on the ability of HDC to provide decomposition over a composite data structure. Fundamental to both perception and cognition, the decomposition problem has been argued to manifest itself in various domains, including visual scene analysis and analogical reasoning [1, 2, 5, 18]. Under the HDC framework, this problem was first modeled as the high-dimensional vector factorization problem [5, 15]: given a hypervector that is the binding of multiple atomic hypervectors, we want to find all its constituents with only the codebooks, i.e. the list of elementary hypervectors. Because the product of binding does not preserve similarity with its components, this problem traditionally requires an exhaustive search over all possible combinations of the elementary hypervectors to solve the factorization. The combinatorial nature of this problem makes the search size prohibitively large for exhaustive searches in practice, limiting the utilization of HDC in prac-

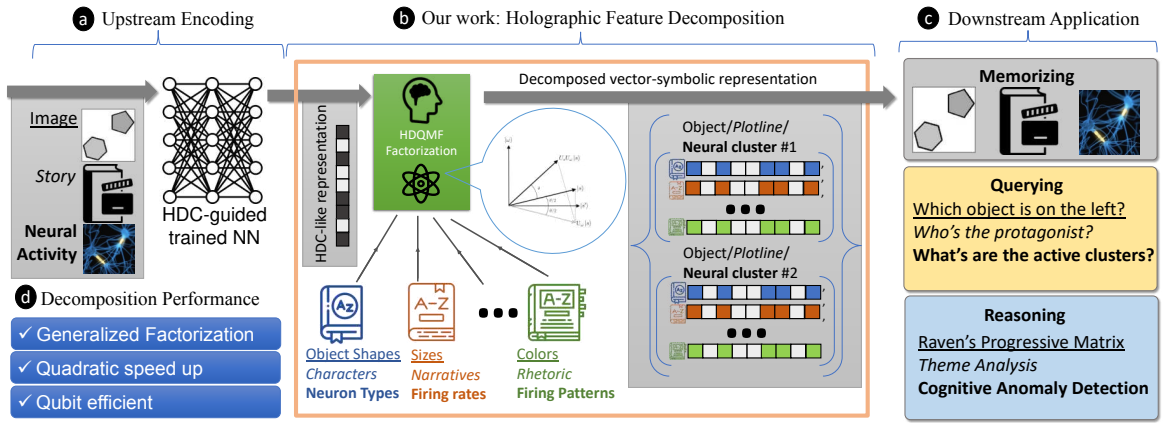

Figure 1. Overview of our approach. (a) The upstream network generates HDC-like holographic representation. (b) HDQMF performs Holographic Feature Decomposition given the HDC-like representation and the salient codebooks. Our approach is inspired by Grover's algorithm. (c) The resulting decomposition can be used for subsequent cognitive tasks, including reasoning. (d) Our approach is specialized for the memorized factorization problem, achieves quadratic speedup asymptotically, and is efficient in qubit usage.

tical scenarios. To the best of our knowledge, all existing models, including [10], compromise by limiting the depth of the data structures or using exhaustive search over all possible combinations when necessary [15]. While effort has been made to optimize the problem classically, given the empirical results and the current limitation of the theoretical analysis tool for this problem [5, 15], we consider it a safe assumption that all classical algorithms have asymptotic time complexity linear in the search space.

Besides the hardness of the problem, the high-dimensional vector factorization problem also requires non-trivial generalization to be applied in typical HDC models. In practice, it is much more common that the hypervectors in focus, like in [10], are a bundling of bound hypervectors. To provide solutions more suitable for this case, we address an extension of the high-dimensional vector factorization problem, called the **HDC Memorized-Factorization Problem**, that captures this additional complexity.

The assumption about the linear time complexity and the added complexity of the factorization problem observed in practice drive us to propose algorithms that can fundamentally pass through the limit imposed by classical computation, which leads us to quantum computing (QC). Quantum computers differ fundamentally from classical computers that use binary bits. Instead, quantum computers encode information in quantum bits, or qubits, that can exist in a superposition of states. Quantum algorithms are well-known to asymptotically speed up certain types of problems, and our factorization problem falls within this domain. In this work, we propose a quantum algorithm for the extended problem with crucial modification to Grover's algorithm, providing quadratic speed-up to classical solutions.

Fig. 1 shows an overview of our work and its general context. Our contributions are as follows:

1. We propose HDQMF, the **H**yper**D**imensional **Q**uantum **M**emorized-**F**actorization algorithm, that addresses the HDC Memorized-Factorization Problem. This algorithm specializes in finding solutions to the extended problem even with the presence of noise.
2. We provide analytical results, circuit-level implementation, and performance measures of our model. We introduce a modified Grover's algorithm suitable for random and approximate search for arbitrary similarity distributions, with analytically tractable results.
3. We propose a hybrid framework for decoding the bundling of bound hypervectors and compare our methods to state-of-the-art approaches. We show our approximate Grover's algorithm retains the quadratic speedup and can be parallelized in a straightforward manner.

2. Background and Related work

2.1. Hyperdimensional Computing

Elementary Hypervectors. Hyperdimensional Computing uses high-dimensional vectors, called hypervectors, to represent symbols. Each class of elementary symbols is typically represented by a codebook, consisting of a list of high-dimensional, robust, holographic, and random hypervectors [14] that preserve the desired measure of similarity, or kernel, between different symbols within the class. In the Multiply-Add-Permute (MAP) scheme [6] where the elementary hypervectors are bipolar, we can impose the Kronecker kernel over a discrete set $\mathcal{D}$ by:

$$\begin{aligned} h_x &\leftarrow \text{Unif}\left\{-1, 1\right\}^D && x \in \mathcal{D} \\ \delta(h_x, h_y) &= x^T y / D && x, y \in \mathcal{D} \end{aligned} \tag{1}$$

Where h_x is the encoding of the data point $x \in \mathcal{D}$, D is the dimension of the hypervector, and $\text{Unif}(\mathcal{S})$ is the uniform

distribution over the support $\mathcal{S}$. The randomizing property of the generation and the high dimensionality ensures that the kernel is well-approximated, such that $\delta(h_x, h_y) \sim 1$ for $x = y$, and $\sim N(0, 1/\sqrt{D})$ otherwise.

HDC Operators. The basic HDC operators are bundling $\oplus$, binding $\odot$, and permutation ρ, which performs "superposition", variable binding, and ordering, respectively. Bundling creates sets such that the resulting hypervector preserves similarity with its constituents [14], while binding functions as an association operator such that the resultant contains the information of both of its constituents but is dissimilar to any of them [4]. In MAP, they are element-wise addition and element-wise multiplication respectively; such implementations align with the functional aspects of the operators. For bundling, the set hypervector $h_S = h_x \oplus h_y \oplus h_z$ for the set $S = \{x, y, z\} \subset \mathcal{D}$ computes set membership efficiently:

$$\delta(h_S, h_w) = \sum_{k \in S} \delta(h_k, h_w) \approx (w \in S) \tag{2}$$

For binding, the bound hypervector, between two or more symbols, $h_{(x,y)} = h_x \odot h_y$, represents associations in that (1) the association hypervector is dissimilar to any of its components, $\delta(h_{(x,y)}, h_x) \approx 0$, and (2) Unbinding one of the atoms from the hypervector reveals the other, $h_{(x,y)} \odot h_x = h_y$.

HDC Schemes. Using the basic HDC operators, one may construct composite hypervectors representing more complex data structures, including trees [5], graphs [29], and finite state automata [20]. One of the most commonly used schemes is the "bind-then-bundle" scheme, where *attributes of each object are bound together, and then multiple objects are bundled to represent the object combination.* Fig. 1(b) implies an instance from such scheme:

$$\begin{aligned} h_{\text{scene}} = h_{hexagon} \odot h_{large} \odot \cdots \odot h_{light} \\ \oplus\, h_{pentagon} \odot h_{large} \odot \cdots \odot h_{dark} \end{aligned} \tag{3}$$

HDC in Action. HDC's benefit lies in its

1. *Representational robustness*: each element of the hypervector contributes evenly to encoding the information (hence holographic) and high-dimensionality ensures that the information is encoded redundantly [14];
2. *Interpretable and fast hypervector manipulation*: every transform by the HDC operator can be interpreted as a certain type of structure manipulation [23, 30], and binding and bundling are element-wise (while circular shift implements permutation); and
3. *Fixed-width composition*: the composite structures constructed from elementary ones with HDC operators, regardless of the schemes, are of fixed dimensionality.

Collectively, HDC becomes a natural candidate for addressing the "binding problems" of neural networks [33] that may provide an adequate description of real-world objects or situations that can be represented by hierarchical and nested compositional structures [10, 31].

To this end, [10] proposes a neural-vector-symbolic architecture consisting of (1) a neuro-vector frontend to generate HDC-like (holographic) representation as perception (Fig. 1(a)-(b)), and (2) a vector-symbolic backend to perform symbolic logic reasoning as cognition (Fig. 1(b)-(c)). The efficient representation of object combination through HDC connects the best of both worlds in solving cognitive tasks. *Most notably, like many other HDC models* [7, 11–13, 16, 19, 21, 22, 24–29, 34, 36], *it uses the "bind-then-bundle" scheme* (Fig. 1(b)).

HDC Decompostion. Many use cases of HDC require one to decompose a composite hypervector back to its elementary components, and the decomposition process requires some knowledge of the codebooks and the schemes. If the codebooks are known, bundling and permutation have clear arithmetic approaches for decoding, as bundling preserves the similarity of its components, and permutation can be iteratively reversed. Since bound hypervectors are dissimilar to their components, the problem becomes computationally harder. [5, 15], the *state-of-the-art*, formalized the high-dimensional vector factorization problem to decompose bound vectors and developed a recurrent network called the resonator network that showed practical advantages in decomposing a bound hypervector over optimization-based approaches. Recent work has proposed the problem of decomposing "bound-then-bundled" hypervectors and applied resonator network to study its effectiveness [9, 32]. In this work, we formulate this problem again and design an algorithm particularly suited for the decomposition.

2.2. Quantum Computing

Quantum Computing Basics. Quantum computing is founded on the principle of applying various unitary operators to manipulate a set of wavefunctions, typically constructed using a class of two-level systems known as qubits which enables quantum computers to represent information as a superposition across all possible states.

A general quantum circuit is designed to implement a unitary operator denoted as U_f within the Hilbert space. This operator represents a binary function, f, mapping inputs from the set $\{0,1\}^m$ to outputs in $\{0,1\}^n$. *The capacity to perform computations in superposition on any set of states empowers quantum computing to potentially offer significant speedup advantages over classical computers in various problem-solving scenarios.* In this work, we rely on multi-control gates such as the multi-control XOR (MXOR) to implement the oracle. The MXOR gate computes the XOR of the components of the n-dimensional qubit state $|\vec{x}\rangle$: $\text{MXOR}\,|\vec{x}\rangle\,|d\rangle = |\vec{x}\rangle\,|d \oplus (\oplus_i x_i)\rangle$.

Grover's Algorithm. Grover's algorithm aims to find a specific item, often referred to as the target, within an unsorted database of size S. It can be summarised as follows:

1. Construct the initial superposition of states given by $|\mathcal{A}\rangle$
2. Apply the oracle operator to U_o, to get $U_o|\mathcal{A}\rangle$
3. Apply the diffusion operator U_D, to get $U_D U_o|\mathcal{A}\rangle$
4. Repeat the previous two steps $\frac{\pi}{4}\sqrt{S}$ number of times, and perform the measurements

The state preparation algorithm builds the operator $U_{\mathcal{A}}$ which generates a uniform superposition of all the possible states $\mathcal{A}$: $U_{\mathcal{A}}\left|\vec{0}\right\rangle = \sum_{a\in\mathcal{A}} \frac{1}{\sqrt{S}}|a\rangle = |\mathcal{A}\rangle$. The Oracle is the unitary operation that searches for the marked state (which we shall denote $|m\rangle$). It simply inverts the phase of the marked state with $U_o|m\rangle = -|m\rangle$, and $U_o|a\rangle = |a\rangle$ otherwise:

$$U_o\left(\frac{1}{\sqrt{S}}\sum_{a\in\mathcal{A}}|a\rangle\right) = \frac{1}{\sqrt{S}}\sum_{a\in\mathcal{A}}(-1)^{\delta_{m,a}}|a\rangle\,. \quad (4)$$

The diffusion operator U_D inverts the amplitudes of the states in the codebook by the average. It is represented by the operator U_D, $U_D = 2|\mathcal{A}\rangle\langle\mathcal{A}| - I = U_{\mathcal{A}}\left(2\left|\vec{0}\right\rangle\left\langle\vec{0}\right| - I\right)U_{\mathcal{A}}^{\dagger}$, with I the identity operator. The diffusion operator simply inverts the amplitudes about the average amplitude as $U_D\sum_{a\in\mathcal{A}}\psi(a)|a\rangle = \sum_{a\in\mathcal{A}}(\mu-\psi(a))|a\rangle$, where $\mu = \frac{1}{S}\sum_{a\in\mathcal{A}}\psi(a)$ is the average of the amplitudes.

After $\approx \frac{\pi}{4}\sqrt{S}$ iterations, the probability of measuring the marked state increases to a maximum value of about 50%, improving the likelihood of finding the marked state.

3. Memorized-Factorization Problem

The original high-dimensional vector factorization problem problem is as follows [5].

Definition 1 (The Factorization Problem)**.** *Given F codebooks $\mathcal{C}_1,\cdots,\mathcal{C}_F$ and a hypervector c, minimize $\|c-\hat{c}\|$ subject to $c_1\in\mathcal{C}_1,\cdots,c_F\in\mathcal{C}_F$ and $\hat{c} = \bigodot_{i=1}^{F} c_i$.*

We may assume that each codebook $\mathcal{C}_i$ is generated randomly and independently as in the MAP scheme and that the solution is unique. If we assume that all codebooks have N entries. The effective search space of the problem is $\Pi_{i=1}^{F}|\mathcal{C}_i| = O(N^F)$. In other words, it scales exponentially with the number of factors and polynomially with the size of the codebook, which makes the problem computationally challenging.

As mentioned in Sec. 2.1, applications of associative binding operations are typically combined with the operation of bundling to enable memorizing multiple items with various attributes, where each set contains elements associated with each other. As we will show, our quantum solution for approximate factorization also enables a direct application toward decoding memorized-associative hypervectors aka "bound-then-bundled" hypervectors. We formally define the Memorized-Factorization Problem as follows:

Definition 2 (The HDC Memorized-Factorization Problem)**.** *Given F codebooks $\{\mathcal{C}_i\}_{i=1}^{F}$, an integer $k>0$, and a vector c, minimize $\|c-\hat{c}\|$, where $\hat{c} = \bigoplus_{j=1}^{k}\bigodot_{i=1}^{F} c_{i,j}$, s.t. $c_{i,j}\in\mathcal{C}_i, i=1,\cdots,F, j=1,\cdots,k$.*

4. HyperDimensional Quantum Memorized-Factorization Algorithm

In this section, we introduce HDQMF which leverages the power of quantum computing to solve the factorization problem in an efficient and approximate manner. HDQMF uses a modified form of Grover's algorithm [8] to search through all possible factorizations, and extract the most accurate one through amplification. Moreover, we show that our design is easily scalable, and, due to the additive nature of the quantum phase, we are able to extend our design to solve the more general memorized-factorization problem in an efficient manner with minimal modification.

From HDC to Quantum. In HDQMF, we represent each element of our hypervector by a two-component qubit state: $|0\rangle$ will represent 1 and $|1\rangle$ will represent -1. The multiplication of the components of the vector will be represented by the XOR of the labels of the corresponding states representing the hypervectors. We represent the $D-$dimensional hypervector h_i by the corresponding qubit series $\left|q_0^i q_1^i ....q_{D-1}^i\right\rangle \equiv \left|\vec{q^i}\right\rangle$. We can store multiple collections of hypervectors as $\left|\vec{q^1}\right\rangle\left|\vec{q^2}\right\rangle\cdots|\vec{q^n}\rangle$, which requires a total of $m\times D$ qubits to represent the m hypervector factors. The binding of two HDC vectors $h_i\odot h_j$ is represented by $\left|\vec{q^i}\oplus\vec{q^j}\right\rangle \equiv \left|(q_1^i\oplus q_1^j)(q_2^i\oplus q_2^j)....(q_{D-1}^i\oplus q_{D-1}^j)\right\rangle$. The Oracle will need to implement the binding operation as part of its overall calculations. We will demonstrate a method to perform this that requires a number of gates linear to the dimension. The initialization of the qubits to lie in all possible states defined by the codebook can be done using a polynomial number of circuits and qubits through the algorithm described in [35]. We impose that the factors belong to the codebook $M_{F\times N\times D}$, where F is the number of factors, N is the number of candidate vectors for each factor, and D is the dimension of the hypervector. Defining $\vec{q}_f^n$ to be the n^{th} vector at the f^{th} factor, the goal of the state preparation algorithm is to construct the following state: $U_{\mathcal{M}}\left|\vec{0}\right\rangle\cdots\left|\vec{0}\right\rangle = \frac{1}{\sqrt{N^F}}\left(\sum_n|\vec{q}_1^n\rangle\right)\cdots\left(\sum_n|\vec{q}_F^n\rangle\right)$, which represent all possible factorizations. This is implemented sequentially, by applying the algorithm of [35] to each factor individually by defining the operator $U_{\mathcal{M}}^k\left|\vec{0}\right\rangle = \frac{1}{\sqrt{N}}\left(\sum_n|\vec{q}_k^n\rangle\right)$, which constructs a uniform superposition of all vectors belonging

to the k^{th} factor of the codebook. By defining the tensor product operator $U_{\mathcal{M}} = \Pi_k \otimes U^k_{\mathcal{M}}$, where each term acts on a different set of qubits as

$$U_{\mathcal{M}} \left|\vec{0}\right\rangle \cdots \left|\vec{0}\right\rangle = \left(U^1_{\mathcal{M}} \left|\vec{0}\right\rangle\right) \left(U^2_{\mathcal{M}} \left|\vec{0}\right\rangle\right) \cdots \left(U^F_{\mathcal{M}} \left|\vec{0}\right\rangle\right), \tag{5}$$

we can generate a uniform superposition of all factors with a polynomial circuit complexity and qubit requirement.

Key Modification. In our application of Grover's algorithm, the marked state will not have an exact phase of -1 applied by the oracle. Rather, the phase shift will be related to the similarity $\delta \in [0, 1]$ with a "true state" which is calculated by the oracle, and the phase shift is $e^{i\pi\delta}$. $\delta = 1$ corresponds to an exact match and $\delta = 0$ to random noise. With such an oracle, we modify the algorithm to:

1. Construct the initial superposition of states given by $|\mathcal{A}\rangle$
2. **Apply either U_o or U_o^{-1} in alternate iterations, starting with U_o^{-1}**
3. Apply the diffusion operator U_D
4. Repeat the previous two steps $\frac{\pi}{4}\sqrt{S}$ number of times, and perform the measurements

In the limit of real phase with $\delta = 0$ or 1, our algorithm reduces to the original Grover's algorithm. As we will see, the application of the inverse U_o operator in alternate steps results in iterative phase cancellation, allowing an exact analytical description of the evolution of the amplitudes with *any* similarity distribution. This allows us to show that the algorithm behaves similarly to the original Grover algorithm, where the efficiency of the algorithm (in terms of the optimal iterations required) can be related to the distribution of phases.

4.1. Approximate Factorization Algorithm

The key aspect of Grover's algorithm is the oracle, which identifies the "marked" states, and inverts their phase to -1. The difficulty in exact quantum search is the binary decision for the oracle, requiring a case-by-case check for whether the output satisfies some specific pattern.

In our design, we exploit the randomness of hypervectors to approximately distinguish the marked and unmarked states. In our design, the phase inversion will not be an exact -1 for the marked states. Instead, the oracle multiplies each state by a phase factor $e^{i\pi\phi}$, where ϕ is the similarity between the multiplied factors $\hat{c}$ and the given vector $\vec{c}$. For relatively large dimensions, $\phi \sim 0$ for randomly chosen factorizations, while $\phi \sim 1$ for correct factorizations. We expect this algorithm to work well because the number of hypervectors that are highly similar is exponentially small compared to the whole space of hypervectors. Thus, the standard deviation of ϕ is expected to be small. With a small spread in the similarity, we expect our algorithm to efficiently extract a factorization that is likely correct.

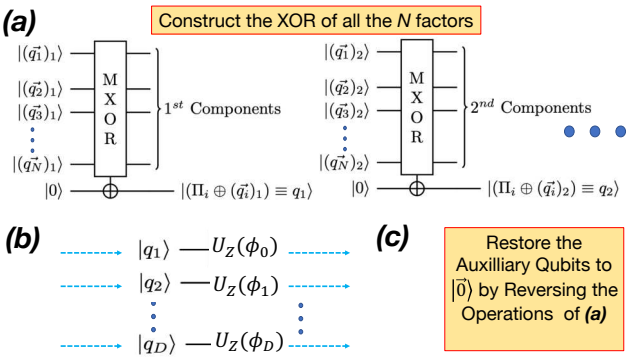

Figure 2. The design of the oracle, involving (a) constructing the binding of the vectors, (b) performing the phase similarity operation, and (c) uncomputing the bound vector.

One key property of the quantum phase is that even if the operation is applied to a certain subset of qubits, the phase shift $e^{i\pi\phi}$ is nonetheless applied to the whole system. Thus, if an operation 1 (resulting in phase $e^{i\pi\phi_1}$) is applied to one subset of qubits, and another operation 2 (resulting in phase $e^{i\pi\phi_2}$), then the system accumulates a phase $e^{i\pi(\phi_1+\phi_2)}$. This additive global property of the quantum phase is exploited to generalize our search algorithm for the memorized-factorization problem. Suppose we have M quantum systems running in parallel, with each system representing one of the possible factorizations. A generic state in this system can be written as $|S_1\rangle |S_2\rangle \cdots |S_M\rangle$, with $|S_i\rangle$ representing a possible factorization for one of the elements v_j bundled in $c = \bigoplus_{j=1}^{k} v_j$, with $v_j = \bigodot_{i=1}^{F} c_{i,j}$. Our oracle is designed to implement $U_o |S_i\rangle = e^{i\theta\pi\delta(c,v_i)} |S_i\rangle$, where the product of factors represented by $|S_i\rangle$ is v_i, and θ is a scale factor to be fixed later. Applying the oracle to each $|S_i\rangle$ sequentially in the whole system results in the phase,

$$U_o \left(\Pi_i |S_i\rangle\right) = \Pi_i U_o |S_i\rangle = e^{i\theta\pi \sum \phi_i} \Pi_i |S_i\rangle . \tag{6}$$

Thus, the total phase accumulated now is $\phi = \theta(\phi_1 + \phi_2 + \cdots + \phi_M)$. If all the M states $|S_i\rangle$ represent factorizations of some v_j, then the total phase is $\phi = 1$. However, unlike the previous algorithm, there are now various cases for the phase, since it could be that only some $m < M$ states represent correct factorizations, and the resulting phase would be $\phi = \frac{m}{M} + \text{Noise}$, where Noise is a noise term from some (approximately gaussian) distribution. We thus have a cascading levels of phase similarity ϕ, taking values $\phi = \theta i$ for $i = 0, 1, 2, .., M$.

4.2. Oracle Design

The key aspect of the approximate search algorithm is to have an approximate oracle that can identify different states based on their similarity. Given a state $|S\rangle = |c_1\rangle \cdots |c_F\rangle$,

representing a specific factorization, the oracle needs to calculate the product $\hat{c} = \bigodot_{i=1}^{F} c_i$, and then calculate the similarity ϕ with a target vector v such that there is a phase change of $e^{i\pi\phi}$. The first step of multiplying the factors to construct $\hat{c}$ is performed using the MXOR gate which enables multiplication with a minimum number of gates. This required D auxiliary qubits to store the output (which must be later uncomputed). We show this design in Fig. 2(a).

Next, the oracle needs to induce a phase change using the similarity of $\hat{c}$ with a given target vector v. The similarity function is given by $\delta(\hat{c}, v) = \frac{1}{D}\sum_{j=1}^{D}(\hat{b})_j(v)_j$, where j indexes the components of the vectors, and $\hat{b} = 1 - 2\hat{c}$ is the bipolar $(1, -1)$ representation. Our design of the oracle exploits this additivity of the similarity to enable an efficient phase transformation, which is illustrated in Fig. 2(b). Moreover, the components of $\hat{c}$ are either 0 or 1, while the vector v can be any generic real number. Thus, the oracle simply applies Pauli Z-gate $U(\phi_j)$ in sequence on each qubit representing the component of $(\hat{c})_j$, with angle $\phi_j = -\pi\frac{(v)_i}{D}$. The gate $U(\phi)$ multiplies the state $|c\rangle$ with a phase factor $e^{i\phi(2c-1)}|c\rangle$, for $c = 0, 1$. Since $(\hat{c})_j$ is either 1 or 0, with phase change $e^{-i\pi\frac{(v)_j}{D}}$ or $e^{i\pi\frac{(v)_j}{D}}$ respectively, the phase shift is simply $e^{i\pi\frac{(v)_j(\hat{b})_j}{D}}$. Thus, after applying this phase operator to all the qubits, the total phase accumulated will be $\Pi_{j=1}^{D} e^{i\pi\frac{(v)_j(\hat{b})_j}{D}} = e^{i\pi\delta(\hat{c},v)}$, as required by the oracle.

If we had multiple, say M factorizations, that needed to be bundled together in the end, then we simply require the oracle to apply the phase shift operation $U(\phi_j)$ with $\phi_j = -\theta\pi\frac{(v)_j}{D}$ on all the qubits, where θ is a scale factor which we will discuss later. The resulting overall phase accumulated will be $\Pi_{j=1}^{M} e^{i\theta\pi\delta(\hat{c}_j,v)} = e^{i\theta\pi\sum_{i=1}^{M}\delta(\hat{c}_i,v)}$. This design of the oracle is scalable since it requires only the application of local phase shifts on the individual qubits. Moreover, there is no requirement to maintain entanglement since the oracle does not do a binary check for a marked/unmarked state. As we will show in the next section, the optimal value of θ is $\theta = 1$.

5. Analytical Study

In this section, we analytically investigate the expected behavior of our algorithm. We denote the amplitude of all the states by the vector $\vec{\alpha}_n$, where n is the iteration of the algorithm, and each component denotes the amplitude of a basis state. We also define $\mu_n = \sum_i(\vec{\alpha}_n)_i/N^{FM}$ to be the average value of the components, where N^{FM} is the total number of states. Additionally, let $\vec{\phi}$ be the similarity associated with each state. We initialize the amplitude to $\vec{\alpha}_1 = \frac{1}{\sqrt{N^{FM}}}\vec{1}$, where $\vec{1}$ is a vector with all components 1. An iteration of the original Grover's algorithm evolves the amplitude as $\vec{\alpha}_{n+1} = (2\mu_n - \vec{\alpha}_n)\cdot\vec{\phi}$. In

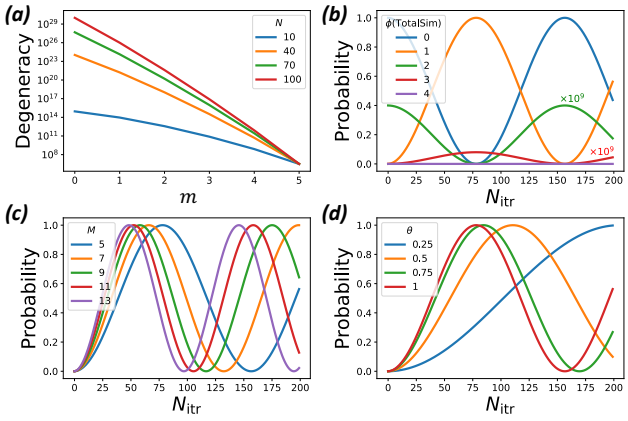

Figure 3. Noiseless results in the $D \to \infty$ limit (a) the exponential decay in the degeneracy for each m (b) The probability as a function of iteration, showing the system is effectively governed by $m = 0$ and $m = 1$ (c) as M increases, then the peak probability is achieved at a lower iteration (d) The optimal value of $\theta = 1$ which requires the lowest number of iterations for the peak probability. We use $N = 100, F = 3, \theta = 1, K = 20, M = 5$ unless mentioned otherwise.

our modification, however, the applied phase transformation changes sign in each step, with the transformations $\vec{\alpha}_{2n+i} = (2\mu_{2n+i-1} - \vec{\alpha}_{2n+i-1})\cdot\vec{\phi}^{2i-1}$ for $i = 0, 1$. By expanding the recurrence relation, we find

$$\vec{\alpha}_{2n} = 2\mu_{2n-1}\vec{\phi}^* - 2\mu_{2n-2} + \vec{\alpha}_{2n-2}, \tag{7}$$

$$\vec{\alpha}_{2n-1} = 2\mu_{2n-2}\vec{\phi} - 2\mu_{2n-3} + \vec{\alpha}_{2n-3}, \tag{8}$$

where we used the fact that $\phi^*\cdot\phi = \vec{1}$, since each component of ϕ is a phase value. We first find the mean amplitude by averaging over the components to find,

$$\begin{pmatrix}\mu_{2n}\\ \mu_{2n-1}\end{pmatrix} = G\begin{pmatrix}\mu_{2n-2}\\ \mu_{2n-1}\end{pmatrix}, G = \begin{pmatrix}4|\lambda|^2 - 1 & -2\lambda^*\\ 2\lambda & -1\end{pmatrix},$$

where λ is the average component of $\vec{\phi}$, which can be written as $\lambda = \sqrt{N^{FM}}\mu_2^*$ (Since $\vec{\alpha}_2 = \frac{1}{\sqrt{N^{FM}}}\vec{\phi}^*$). Note that in the case of the traditional Grover's algorithm, $\lambda = 1 - 2r$, where r is the fraction of the marked state over all the states.

The eigenvalues of G are given by $\gamma_\pm = -1 + 2|\lambda|^2 \pm 2|\lambda|i\sqrt{1 - |\lambda|^2}$. Since they satisfy $|\gamma_\pm| = 1$, the eigenvalues are phase values. The corresponding eigenvectors are $\vec{v}_\pm = (|\lambda|^2 \pm i|\lambda|\sqrt{1 - |\lambda|^2}, \lambda) = (\frac{\gamma_\pm + 1}{2}, \lambda)$. Solving the recurrence equations, we find the mean value to transform:

$$\begin{pmatrix}\mu_{2n}\\ \mu_{2n-1}\end{pmatrix} = \frac{1}{2\lambda\sqrt{N^{FM}}}\sum_{\sigma=\pm}\gamma_\sigma^{n-1}\vec{v}_\sigma = \vec{\mu}_n. \tag{9}$$

By substituting this into the recurrence relationship, we can find a closed form for $\vec{\alpha}_{2n}, \vec{\alpha}_{2n-1}$. Defining $\gamma_\pm = e^{\pm i\theta}$,

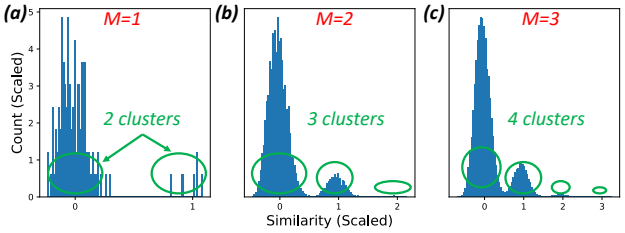

Figure 4. Distribution of similarities for various M, which show $M+1$ clusters, one for each value of similarity. The degeneracy decays with increasing similarity.

$a_{n-1} = \cos\left(\frac{n-2}{2}\theta\right)\frac{\sin\left(\frac{n-1}{2}\theta\right)}{\sin\frac{\theta}{2}}$, $b_n = (2a_n - 1)/\sqrt{N^{FM}}$, and $c_n = (a_n + a_{n-1} - 1)/(\lambda\sqrt{N^{FM}})$ we have

$$\vec{\alpha}_{2n} = \vec{\phi}^* b_n - c_n, \quad \vec{\alpha}_{2n-1} = \vec{\phi} c_n - b_n. \tag{10}$$

Using these results, we can find that the approximate number of iterations required for Grover's algorithm to reach the optimal state as $N_{\text{itr}} \approx \pi/(2\sqrt{2(1-|\lambda|)})$ in the limit of $|\lambda| \lesssim 1$ (which is what we will mostly deal with in our use cases). Note that in the original Grover's algorithm, $\lambda = (A-B)/(A+B)$, where A is the number of unmarked states (with phase 1) and B is the number of marked states (with phase -1), and in the limit of $B \ll A$ we recover the original result of $N_{\text{itr}} \approx \frac{\pi}{4}\sqrt{\frac{A+B}{B}}$.

In order for the algorithm to efficiently identify the marked state with a high probability, we need to minimize the value of N_{itr}, which is done by minimizing the value of $|\lambda|$. In our application, we can calculate the exact value of λ in the $D \to \infty$ limit (where the similarities are either 1 or 0, without noise). When M vectors are bundled together, m of them can match with the bundled set $c = \oplus_{j=1}^{k} v_j$, and the remaining $M-m$ do not. If they match, then they can be one of the k possible vectors, and if they do not then they can be one of the remaining $N^F - k$ vectors. Thus, the total number of such arrangements is given by $k^m(N^F-k)^{M-m}\binom{M}{m}$. The corresponding phase factor is given by $e^{i\theta\pi m}$. Thus, we find $\lambda = \frac{\sum_m \left(e^{i\theta\pi}k\right)^m (N^F-k)^{M-m}\binom{M}{m}}{\sum_m k^m (N^F-k)^{M-m}\binom{M}{m}} = \frac{\left((e^{i\theta\pi}-1)k+N^F\right)^M}{N^{MF}}$. From this expression, we can deduce that the optimal number of iterations (minimum $|\lambda|$) is achieved when $\theta = 1$ (so that $e^{i\theta\pi} = -1$). In this limit, λ can be approximately found to be (when $N^F \gg k$) $\lambda = \frac{\left(N^F-2k\right)^M}{N^{MF}} \approx 1 - \frac{2kM}{N^F}$. The optimal number of iterations is thus $N_{\text{itr}} = \frac{\pi}{4}\sqrt{\frac{N^F}{kM}}$.

6. Results

We show simulated results of the approximate Grover's algorithm for the factorization and memorized factorization. The primary parameters we need to vary are the number of factors F, the number of terms to be bundled together K, the number of codebook entries N, and the dimension D. We note that the factorization problem is a specialized case of the memorized-factorization problem with $K = 1$. A comparison with the SOTA, the resonator network, is shown in Tab. 1, highlighting the efficiency of HDQMF.

The effect of the various parameters on the efficiency of the algorithm is primarily through the similarity and the level of noise in similarity introduced by the oracle. We assume that each property being bound together results in a

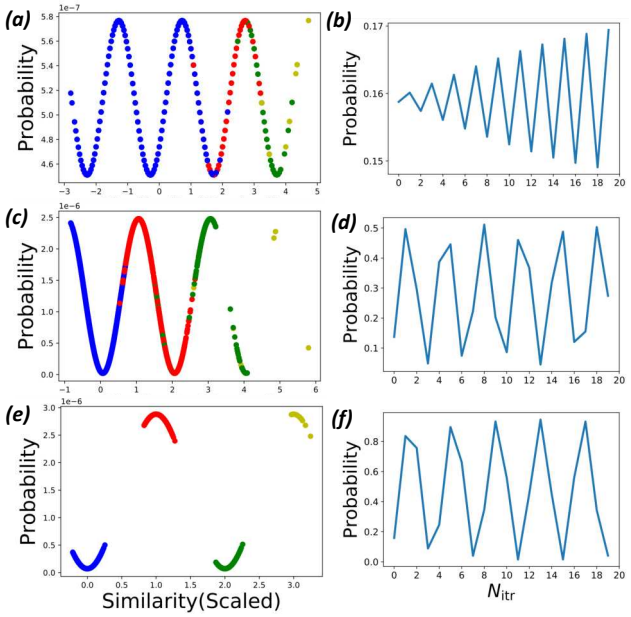

Figure 5. The correlation of probability and similarity after one iteration (left), and the evolution of the probability of marked state (right) for $D = 50, 500, 5000$ (top, middle, bottom). Blue, red, green, and yellow mark $m = 0, 1, 2$ and 3 respectively. As D increases, the similarity values get more clustered and the maximum probability of finding a marked state also increases drastically. We use $F = 3$, $N = 5$, and $K = 7$.

(D, K)	Resonator Network [5]				HDQMF		
	P_s	N_I	P_f	N_S	P_s	N_I	N_S
$(50, 10)$	0.06	20	0.15	338	0.025	**2**	**72**
$(50, 20)$	0.07	26	0.12	368	0.017	**2**	**91**
$(500, 10)$	0.10	34	0.19	338	0.064	**3**	**44**
$(500, 20)$	0.18	35	0.18	195	0.041	**2**	**47**

Table 1. Comparison of quantum search with resonator network, highlighting the optimality of quantum search. P_s is the maximum probability of finding the correct factorization for quantum search and the probability it converges to the correct factorization for the resonator network. N_I is number of iterations to converge/reach P_s. $N_S = N_I/P_s$ is the effective number of steps required. P_f is the probability of non-convergence for the resonator network.

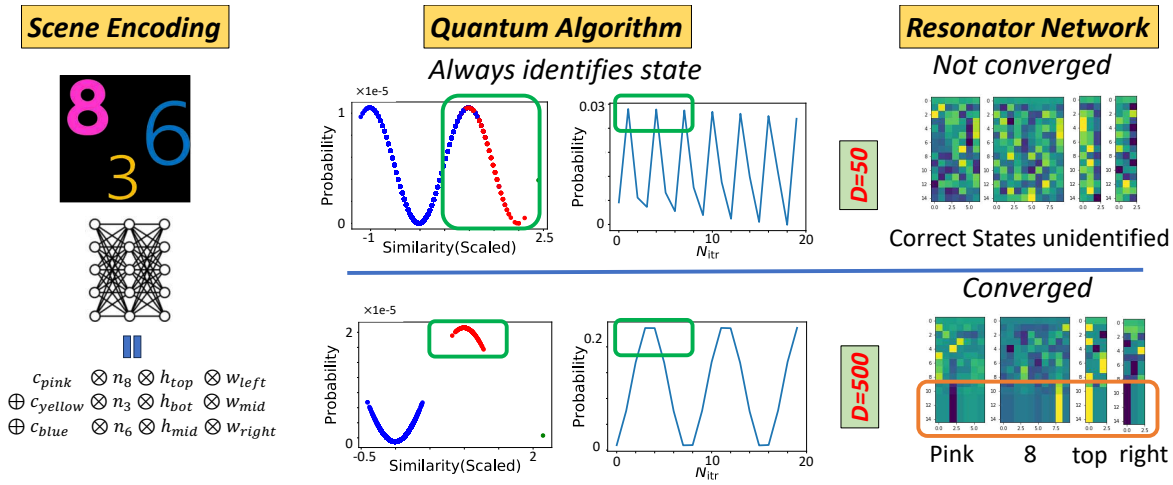

Figure 6. Memorizing a scene with numbers, colors, and locations. The bundled vector is first generated (using alternate means, for example a neural network), and then decomposed using both the quantum algorithm and the resonator network. With $D = 50$ (top), the resonator network does not converge while the quantum algorithm correctly identifies the marked state. At $D = 500$ (bottom), the quantum algorithm and resonator network correctly identify the marked state, but the resonator network requires more iterations.

unique random vector representing the object, which results in N^F different combinations possible for the search space in each object being memorized. If there are K different items, then the noise in the similarity will increase with K if there are mismatches. This noise is also related to that due to M, which is the number of bundled vector being considered in the algorithm to perform the search operation. In the worst-case scenario where there are no matches, the noise is proportional to $\sqrt{KM/D}$.

Noiseless case. We first show the noiseless case of the probability distribution evolution over the iteration. Essentially, this involves taking the limit of $D \to \infty$, and the similarity now depends only on M. We show various cases of M in Fig. 3. (a) shows the degeneracy of each state, decreasing exponentially in m, the number of matching terms. (b) shows the iterations of the probability for each similarity ϕ in a system with $M = 4$. As we can see, the $2, 3$ and 4 matches have probabilities many orders of magnitude smaller than the probability of 0 or 1 match. In (c), we show that by increasing M, the maximum probability for the marked states is reached at smaller iterations, as expected from our theoretical understanding. Finally, we demonstrate the optimality of $\theta = 1$ in (d), which shows that the maximum probability is achieved for the smallest N_{itr} at $\theta = 1$.

Noisy case. Next, we analyze the impact of noise by generating the similarity directly from a randomly sampled codebook. We show the similarity distribution in Fig. 4, for $M = 1, 2, 3$. The distribution is a combination of $M + 1$ Gaussians, centered upon each integer similarities. The width of the Gaussian is proportional to $\sqrt{kM/D}$. In Fig. 5, we show a correlation between the probability and similarity, and the probability of finding a marked state as a function of iterations, for various dimensions $D = 50, 500, 5000$ (from top to bottom). As D increases, there are two effects: (i) the similarity gets more clustered together, with the probability getting larger and (ii) the peak probability at optimal iteration increases drastically.

Use case demonstration. In Fig. 6, we show a case study where we memorize a set of numbers in a scene. The numbers can be 0 to 9, with 7 possible colors, 3 possible vertical locations and 3 horizontal locations. With dimensions $D = 50$ and $D = 500$ the quantum algorithm correctly decomposes the bundled representation within a few iterations. The resonator network, however, does not converge for $D = 50$, and often fails in the case of $D = 500$, while requiring more iterations.

7. Discussion and Conclusion

Limitations. While the algorithm speeds up the hypervector decomposition, the practicality of HDQMF relies on the quality of quantum computers. As quantum technology is still developing, there are limitations to the number of qubits available, the stability of quantum states, and the overall reliability of quantum computations. Our approach will likely be used in conjunction with classical solutions.

Conclusion. We investigated the HDC Memorized-Factorization Problem, which captures the common construction patterns of hypervectors. We proposed HDQMF, based on Grover's algorithms, that addresses this extended factorization problem. We analytically showed and experimentally verified that our algorithm achieves quadratic speedup and can be parallelized straightforwardly.

References

[1] Alexander G Anderson, Kavitha Ratnam, Austin Roorda, and Bruno A Olshausen. High-acuity vision from retinal image motion. *Journal of vision*, 20(7):34–34, 2020. 1

[2] Charles F Cadieu and Bruno A Olshausen. Learning intermediate-level representations of form and motion from natural movies. *Neural computation*, 24(4):827–866, 2012. 1

[3] Jerry A Fodor and Zenon W Pylyshyn. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71, 1988. 1

[4] E Paxon Frady, Denis Kleyko, and Friedrich T Sommer. A theory of sequence indexing and working memory in recurrent neural networks. *Neural Computation*, 30(6):1449–1513, 2018. 3

[5] E Paxon Frady, Spencer J Kent, Bruno A Olshausen, and Friedrich T Sommer. Resonator networks, 1: an efficient solution for factoring high-dimensional, distributed representations of data structures. *Neural computation*, 32(12):2311–2331, 2020. 1, 2, 3, 4, 7

[6] Ross W Gayler. Multiplicative binding, representation operators & analogy (workshop poster). 1998. 2

[7] Ross W Gayler and Simon D Levy. A distributed basis for analogical mapping. In *New Frontiers in Analogy Research; Proc. of 2nd Intern. Analogy Conf*, 2009. 3

[8] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996. 4

[9] Michael Hersche, Zuzanna Opala, Geethan Karunaratne, Abu Sebastian, and Abbas Rahimi. Decoding superpositions of bound symbols represented by distributed representations. In *Proceedings of the 17th International Workshop on Neural-Symbolic Learning and Reasoning (NeSy)*, 2023. 3

[10] Michael Hersche, Mustafa Zeqiri, Luca Benini, Abu Sebastian, and Abbas Rahimi. A neuro-vector-symbolic architecture for solving raven's progressive matrices. *Nature Machine Intelligence*, 5(4):363–375, 2023. 1, 2, 3

[11] Mohsen Imani, Chenyu Huang, Deqian Kong, and Tajana Rosing. Hierarchical hyperdimensional computing for energy efficient classification. In *Proceedings of the 55th Annual Design Automation Conference*, pages 1–6, 2018. 3

[12] Mohsen Imani, Samuel Bosch, Sohum Datta, Sharadhi Ramakrishna, Sahand Salamat, Jan M Rabaey, and Tajana Rosing. Quanthd: A quantization framework for hyperdimensional computing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(10):2268–2278, 2019.

[13] Mohsen Imani, Ali Zakeri, Hanning Chen, TaeHyun Kim, Prathyush Poduval, Hyunsei Lee, Yeseong Kim, Elaheh Sadredini, and Farhad Imani. Neural computation for robust and holographic face detection. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pages 31–36, 2022. 3

[14] Pentti Kanerva. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional vectors. *Cognitive Computation*, 2009. 1, 2, 3

[15] Spencer J Kent, E Paxon Frady, Friedrich T Sommer, and Bruno A Olshausen. Resonator networks, 2: Factorization performance and capacity compared to optimization-based methods. *Neural computation*, 32(12):2332–2388, 2020. 1, 2, 3

[16] Yeseong Kim, Mohsen Imani, and Tajana S Rosing. Efficient human activity recognition using hyperdimensional computing. In *Proceedings of the 8th International Conference on the Internet of Things*, pages 1–6, 2018. 3

[17] 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. 1

[18] Roland Memisevic and Geoffrey E Hinton. Learning to represent spatial transformations with factored higher-order boltzmann machines. *Neural computation*, 22(6):1473–1492, 2010. 1

[19] 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. 3

[20] Evgeny Osipov, Denis Kleyko, and Alexander Legalov. Associative synthesis of finite state automata model of a controlled object with hyperdimensional computing. In *IECON 2017-43rd Annual Conference of the IEEE Industrial Electronics Society*, pages 3276–3281. IEEE, 2017. 3

[21] Evgeny Osipov, Sachin Kahawala, Dilantha Haputhanthri, Thimal Kempitiya, Daswin De Silva, Damminda Alahakoon, and Denis Kleyko. Hyperseed: Unsupervised learning with vector symbolic architectures. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. 3

[22] 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. 3

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

[24] Prathyush Poduval, Mariam Issa, Farhad Imani, Cheng Zhuo, Xunzhao Yin, Hassan Najafi, and Mohsen Imani. Robust in-memory computing with hyperdimensional stochastic representation. In *2021 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, pages 1–6. IEEE, 2021. 3

[25] Prathyush Poduval, Yang Ni, Yeseong Kim, Kai Ni, Raghavan Kumar, Rosario Cammarota, and Mohsen Imani. Hyperdimensional self-learning systems robust to technology noise and bit-flip attacks. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2021.

[26] Prathyush Poduval, Zhuowen Zou, Hassan Najafi, Houman Homayoun, and Mohsen Imani. Stochd: Stochastic hyperdimensional system for efficient and robust learning from raw data. In *IEEE/ACM Design Automation Conference (DAC)*, 2021.

[27] Prathyush Poduval, Zhuowen Zou, Xunzhao Yin, Elaheh Sadredini, and Mohsen Imani. Cognitive correlative encoding for genome sequence matching in hyperdimensional system. In *IEEE/ACM Design Automation Conference (DAC)*, 2021.

[28] Prathyush Poduval, Yang Ni, Yeseong Kim, Kai Ni, Raghavan Kumar, Rossario Cammarota, and Mohsen Imani. Adaptive neural recovery for highly robust brain-like representation. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pages 367–372, 2022.

[29] Prathyush Poduval, Ali Zakeri, Farhad Imani, Haleh Alimohamadi, and Mohsen Imani. Graphd: Graph-based hyperdimensional memorization for brain-like cognitive learning. *Frontiers in Neuroscience*, page 5, 2022. 3

[30] Job Isaias Quiroz-Mercado, Ricardo Barrón-Fernández, and Marco Antonio Ramírez-Salinas. Semantic similarity estimation using vector symbolic architectures. *IEEE Access*, 8: 109120–109132, 2020. 3

[31] Dmitri A Rachkovskij and Ernst M Kussul. Binding and normalization of binary sparse distributed representations by context-dependent thinning. *Neural Computation*, 13(2): 411–452, 2001. 3

[32] Alpha Renner, Lazar Supic, Andreea Danielescu, Giacomo Indiveri, Bruno A Olshausen, Yulia Sandamirskaya, Friedrich T Sommer, and E Paxon Frady. Neuromorphic visual scene understanding with resonator networks. *arXiv preprint arXiv:2208.12880*, 2022. 3

[33] Frank Rosenblatt et al. *Principles of neurodynamics: Perceptrons and the theory of brain mechanisms*. Spartan books Washington, DC, 1962. 3

[34] Anthony Thomas, Sanjoy Dasgupta, and Tajana Rosing. A theoretical perspective on hyperdimensional computing. *Journal of Artificial Intelligence Research*, 72:215–249, 2021. 3

[35] Dan Ventura and Tony Martinez. Initializing the amplitude distribution of a quantum state. *Foundations of Physics Letters*, 12:547–559, 1999. 4

[36] Zhuowen Zou, Hanning Chen, Prathyush Poduval, Yeseong Kim, Mahdi Imani, Elaheh Sadredini, Rosario Cammarota, and Mohsen Imani. Biohd: an efficient genome sequence search platform using hyperdimensional memorization. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*, pages 656–669, 2022. 3