A Brain-Inspired Hierarchical Reasoning Framework for Cognition-Augmented Prosthetic Grasping

Laura I. Galindez Olascoaga, Alisha Menon, Mohamed Ibrahim, Jan M. Rabaey

Berkeley Wireless Research Center, EECS Department, University of California, Berkeley
{laura.galindez, allymenon, mohamed.ibrahim, jan_rabaey}@berkeley.edu

Abstract

This paper proposes a hierarchical framework that enables reasoning across multiple levels of abstraction, from perception to high-level control. The framework relies primarily on Hyperdimensional (HD) computing, a versatile brain-inspired computing paradigm that can produce composite distributed representations for efficient classification and can also facilitate symbolic reasoning. We present this framework through an example use case corresponding to cognition-augmented prosthetic grasping, where user intent is probabilistically predicted to aid the user in successfully selecting and executing the most appropriate gripping mode. Overall, this paper aims to illustrate how HD computing can constitute a mathematical formalism capable of integrating various levels of cognition under a common hierarchical framework.

Introduction

Data-driven artificial intelligence (AI) and machine learning methods have demonstrated great success at perception-based benchmarks, sparking great interest in their adoption within various disciplines, including biomedical engineering, robotics, and edge computing (Park, Took, and Seong 2018; Zhou et al. 2019). Yet, the deployment of AI in practical scenarios may present heterogeneous requirements that range from the inclusion of uncertainty-capturing models of the world to the incorporation of expert knowledge and symbolic reasoning over it. This calls for mathematical formalisms and frameworks that can integrate heterogeneous algorithms and representations.

Furthermore, human cognition is hierarchical by nature (Jeon 2014). At the low level, cognition is established by the integration of massive sensory data from disparate sources. In contrast, high level cognition involves abstract reasoning supported by pre-conceived notions of the world's properties. This hierarchical scheme ultimately drives human *action*, which, in turn, restarts the sense-reason-act closed-loops that brains are uniquely endowed with.

This paper posits that Hyperdimensional (HD) computing (Kanerva 2009), known also as Vector Symbolic Architectures (VSA) (Gayler 2003), can constitute a mathematical formalism capable of integrating the heterogeneous aspects of cognition described above. HD computing builds upon a carefully designed algebra of vectors, that enables symbolic computations as well as the composite encoding of heterogeneous information. Moreover, due to its high-capacity and distributed nature, HD computing is robust against noise (Frady, Kleyko, and Sommer 2018) and can be memory efficient (Zhou, Muller, and Rabaey 2021; Kleyko, Frady, and Sommer 2021).

The posture above is motivated by the multiple successful implementations of HD computing available in literature; earlier on in its development focusing primarily on symbolic reasoning (Plate 1994b; Eliasmith et al. 2012; Rachkovskij and Slipchenko 2012), and more recently demonstrating its usefulness for various perception tasks in the fields of robotics (Neubert, Schubert, and Protzel 2019; Mitrokhin et al. 2019) and bio-signal processing and classification (Rahimi et al. 2018). For perception tasks, in particular, HD computing enables the compact encoding of composite spatio-temporal and frequency distribution structures by exploiting the properties of high-dimensional binary and bipolar vector spaces (Kleyko et al. 2021a). Classification often takes place by means of a simple nearest-centroid search, with the class-centroids also represented as high-dimensional vectors and stored in an *associative memory*. Other approaches follow an iterative learning strategy that improves upon the often sub-optimal nearest-centroid search mentioned before (Diao et al. 2021).

In this paper, we explore the capabilities of HD computing as a formalism to integrate classification, probabilistic inference, and symbolic reasoning within a single computational framework. We envision that this framework can enable reasoning and, eventually, learning across different levels of abstraction by fully describing it in the HD space. The contributions of this paper can be summarized as follows:

Proposal and illustration of a hierarchical reasoning framework. Through a synthetic prosthetic grasping example, we illustrate how the proposed framework can hierarchically represent the different levels of abstraction of the application: at the lowest (perception) level it relies on bio-signal classification; at the intermediate level it relies on a Dynamic Bayesian Network (DBN) to infer user-intent over time; and at the highest level it relies on symbolic reasoning for control and feedback.

Mapping of a probabilistic graphical model to HD space. We demonstrate how to map the DBN mentioned above to the HD space. We show that this mapping allows us to store the conditional probability tables (CPTs) of the model in superposition, with the potential of reducing memory footprint, and enabling an end-to-end HD representation for inference in the proposed framework.

Discussion of the implications and extensions of the proposed framework. We discuss the implications of the current framework and detail future work directions.

Notation

Random variables are denoted with upper case letters X, and their instantiated values with lower case letters x. For long variable and value names, we capitalize the first letter to discern them (e.g. $Task$ is a variable and $task$ a value). Instantiations of multiple variables and vectors are denoted with bold lower case letters $\mathbf{x}$. We distinguish the high-dimensional vectors (or hypervectors, as introduced in the following section) that constitute the primitives of HD computing by denoting them with X.

Hyperdimensional computing

Inspired by the understanding that the brain's computations rely on massive circuits of neurons and synapses, HD computing operates on pseudo-random high-dimensional vectors through a well defined computational algebra of vectors, different from linear algebra (Kanerva 2009). Pairing such a representation and algebra with an associative memory yields a versatile architecture that enables efficient learning and reasoning (Rachkovskij, Kussul, and Baidyk 2013).

Main properties and components

HD computing can materialize through different models, each with their own set of properties. See (Kleyko et al. 2021b) for an extensive list. In general, all HD computing instances rely on the following components and properties (Neubert, Schubert, and Protzel 2019):

High-dimensional spaces. The primitives of HD computing are vectors with n dimensions commonly larger than 10^2, denoted as $\mathsf{X} = \mathbf{x} \in \mathbb{R}^n$. For the rest of the paper, we refer to these high-dimensional vectors as *hypervectors*. Depending on the model, hypervectors can be real-numbered tensors (Smolensky 1990), real and complex valued vectors (Plate 1994a), binary dense (Kanerva et al. 1997) or sparse (Rachkovskij 2001) vectors or bipolar vectors (Gayler 1998). In this paper, we use the Multiply Add Permute (MAP) model (Gayler 1998), which relies on the latter bipolar representation. The choice of high-dimensional spaces is motivated by several of their properties, as explained throughout the following paragraphs, such as the near orthogonality of random hypervectors, their robustness to noise and their huge capacity, even when represented sparsely (Frady, Kleyko, and Sommer 2018).

Orthogonality and similarity metrics. HD computing exploits the properties of high-dimensional spaces. In particular, the hypervectors mentioned above are created randomly by uniformly and independently sampling each dimension from the underlying HD space (Neubert, Schubert, and Protzel 2019). These *seed* hypervectors are often saved in an *item memory* (I.M.), implemented as a code-book or look-up table. Due to their high dimension, two randomly generated seed hypervectors are very likely orthogonal. This results in significant robustness against noise when trying to recognize them or recover them from the I.M. through appropriate distance metrics, such as Hamming distance or cosine similarity (Kleyko et al. 2021b). We refer to the *similarity* between two hypervectors as $\mathrm{sim}(\cdot,\cdot)$. Depending on the metric of choice, similarity is directly (e.g. cosine similarity) or inversely (e.g. Hamming distance) proportional to it. Moreover, due to their approximate orthogonality, seed hypervectors stored in the I.M. can be interpreted and deployed as discrete symbols. This elementary property of high-dimensional spaces constitutes the key of HD computing, and is exploited for the construction of data structures through specialized operations, and for recovering information from noisy inputs as well as for symbolic and sub-symbolic queries through a simple nearest neighbor search, as detailed in the following paragraphs.

HD computing operations. HD computing relies on three main operations:
1) Binding $\circ$: The binding operation combines two input hypervectors into an output hypervector *dissimilar* to the input. The MAP model implements binding as an component-wise product of the input hypervectors. One of the most common uses of the binding operation is to form associations between two concepts or variables in a role-filler or key-value fashion, such as a record describing sensor type $\mathsf{SENSOR} = \mathsf{TYPE} \circ \mathsf{EMG}$. Moreover, each of the original vectors can be recovered by *unbinding* it from the original association. In the MAP model, each vector is its own multiplicative inverse, such that binding a vector to itself results in a hypervector of ones (i.e. $\mathsf{X} \circ \mathsf{X} = \mathbf{1}$, with $\mathsf{X} \in \mathbb{R}^n$ and $\mathbf{1} = (1, 1, ..., 1) \in \mathbb{R}^n$). This self-inverse property means that the unbinding operation is the same as binding, allowing to easily recover the elements of associations. For example, the type of sensor can be recovered from the SENSOR association by unbinding TYPE from it (i.e. $\mathsf{TYPE}\circ\mathsf{SENSOR} = \mathsf{TYPE} \circ \mathsf{TYPE} \circ \mathsf{EMG} = \mathbf{1} \circ \mathsf{EMG} = \mathsf{EMG}$). Binding is associative and commutative and distributes over the bundling operation, explained below.
2) Bundling $+$: The bundling operation, typically implemented as an component-wise sum of hypervectors, combines or *superposes* two inputs into an output that is *similar* to the two inputs. This property results from the near orthogonality of random hypervectors. For example, suppose we have a database $\mathcal{D} = \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$ of random hypervectors and we create the bundle $\mathsf{X} = \mathsf{A} + \mathsf{B}$. Then, $\mathrm{sim}(\mathsf{A}, \mathsf{X}) >> \mathrm{sim}(\mathsf{C}, \mathsf{X})$ and $\mathrm{sim}(\mathsf{B}, \mathsf{X}) >> \mathrm{sim}(\mathsf{C}, \mathsf{X})$ because A and B act symmetrically as noise for the recognition of the other and their noisy versions are more similar to their original versions than to any other random hypervector.
3) Permutation (also called protect) ρ: Permuting a vector produces a dissimilar vector to the original one. This operation is often used to "protect" ordering that could be lost due to the associative and distributive rules of binding. Note that permutations distribute over the previous two operations.

Data structures

Through the components described above, HD computing offers the possibility of encoding data structures into compound hypervectors (Kleyko et al. 2021a), for example:

1) Key-value pairs: Binding can be used to represent associations between elements as key-value or variable-value pairs. For example, in the SENSOR association TYPE is the key and EMG the value.

2) Sets and histograms: Sets can be represented through hypervector bundling. For example, the set $S = \{a, b, c\}$ can be represented as the bundle $\mathsf{S} = \mathsf{A} + \mathsf{B} + \mathsf{C}$. For multi-sets, such as $H = \{a, a, b, c, c, c\}$, a frequency distribution can be represented through the bundle $\mathsf{H} = 2\mathsf{A} + \mathsf{B} + 3\mathsf{C}$. The frequency of a specific hypervector, can be approximately recovered from a bundle through their dot product, which we denote as $\mathrm{dot}(\cdot, \cdot)$. For example, the frequency of A in H can be calculated as $\mathrm{dot}(\mathsf{H}, \mathsf{A})/n \approx 2$, where n denotes the hypervectors' dimension.

3) Sequences: The ordering of the components in a sequence can be preserved through the permutation operation ρ. As such, the sequence $Z = (a, b, c)$ can be represented in HD space as $\mathsf{Z} = \rho^2(\mathsf{A}) + \rho^1(\mathsf{B}) + \rho^0(\mathsf{C})$, where $\rho^m(\mathsf{V})$ denotes that hypervector V is permuted m times.[1]

Parsing representations and queries

The data structures above can be combined to create compound representations (Kleyko et al. 2021a). For example, the record listing the characteristics of a specific prosthetic device can be specified in terms of its sensors and the location (or context) where it is normally used. For example, $\mathsf{DEV} = \mathsf{SENSOR} + \mathsf{CONTEXT}$, where $\mathsf{SENSOR} = \mathsf{TYPE_{SENS}} \circ \mathsf{EMG}$ and $\mathsf{CONTEXT} = \mathsf{LOC} \circ \mathsf{HOME}$. Thanks to the properties described above, we can, for example, probe the record to determine the location where the device is used by unbinding LOC from DEV:

$$\begin{aligned}\mathsf{DEV} \circ \mathsf{LOC} &= (\mathsf{TYPE_{SENS}} \circ \mathsf{EMG} + \mathsf{LOC} \circ \mathsf{HOME}) \circ \mathsf{LOC} \\ &= \mathsf{TYPE_{SENS}} \circ \mathsf{EMG} \circ \mathsf{LOC} + \mathsf{LOC} \circ \mathsf{HOME} \circ \mathsf{LOC} \\ &= noise + \mathsf{HOME} \approx \mathsf{HOME}.\end{aligned}$$

Since $\mathsf{TYPE_{SENS}} \circ \mathsf{EMG} \circ \mathsf{LOC}$ is dissimilar to any of the seed hypervectors in I.M., it can be regarded as noise. The HOME hypervector is then recovered by selecting the neighbor nearest to $\mathsf{HOME} + noise$ in the I.M. Thus, the probing query "cleans up" the noise of the compound representation.

Proposed hierarchical framework

The proposed hierarchical framework is illustrated in Figure 1 and corresponds to a *cognition-augmented* prosthetic grasping use case.

Use case description

This synthetic use case is intended to augment the EMG-based gesture recognition application in (Moin et al. 2021) and (Zhou, Muller, and Rabaey 2021). The goal of these works was to classify EMG (and accelerometer in the latter) data into hand-gestures, which can be used by prosthesis users to select the appropriate *grip* (Figure 3.b shows an

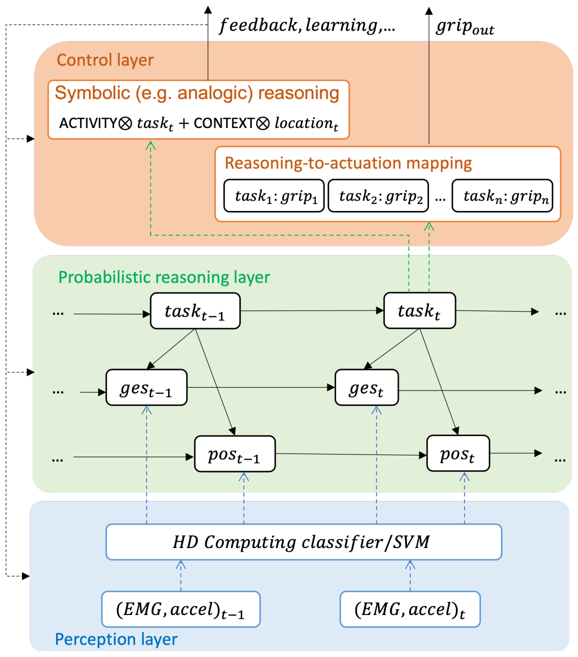


Figure 1: Proposed framework.

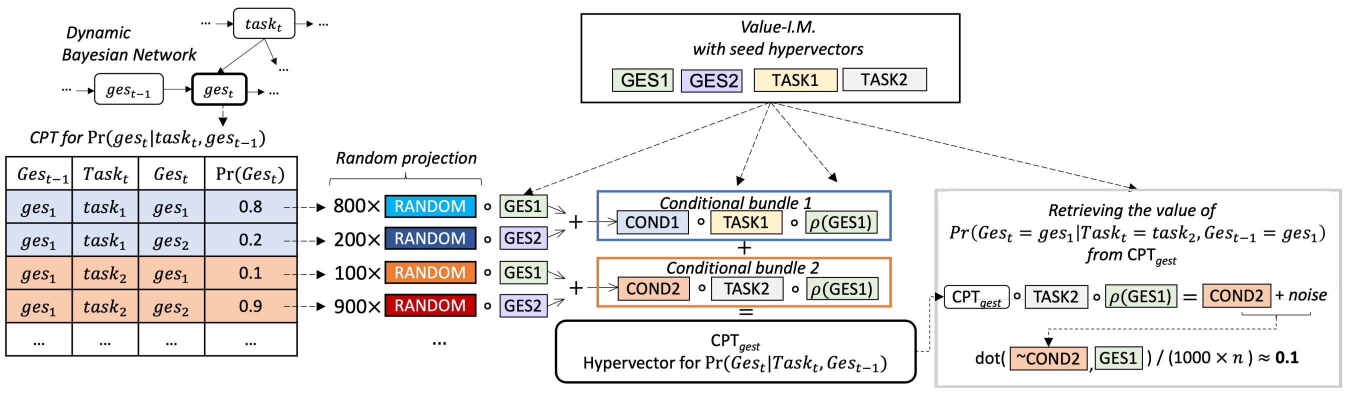
example of these mappings). Our proposed use case is motivated by the observation that prosthesis users often struggle to remember and select appropriate gesture-to-grip mappings (Espinosa and Nathan-Roberts 2019). We propose to augment this application by incorporating user-intent prediction, which can ultimately aid prosthesis users through grip selection and deployment. Moreover, as described later on, the higher level specification of user intent can be used for analogical reasoning towards adaptation to new contexts.

Hierarchical reasoning framework

The proposed framework comprises three layers:

1) Perception layer. The perception layer, highlighted in blue, maps heterogeneous sensory inputs to higher level descriptive concepts, such as hand-gesture and arm-position. Therefore, for this layer, we reuse the representations and models proposed in (Zhou, Muller, and Rabaey 2021). EMG signals are windowed over 50 millisecond overlapping sections, and are mapped to spatio-temporal hypervector representations. The input "query" hypervectors are then mapped to specific gestures through a nearest centroid classifier. The centroids in the model are specified as class-prototype hypervectors, and classification takes place by selecting the class that maximizes the (cosine) similarity between its prototype and the query vector. Accelerometer data is used to classify arm-position through a Support Vector Machine (SVM). In (Zhou, Muller, and Rabaey 2021), position information is used to improve gesture classification performance through dual-stage classification. In this work, the classified positions and gestures are both fed to the following layer.

[1] In practice, permutation commonly consists of cyclic shifts.


Dynamic Bayesian Network
$task_t$
ges_{t-1}
ges_t
CPT for $\Pr(ges_t|task_t, ges_{t-1})$
Random projection
800× RANDOM ∘ GES1
200× RANDOM ∘ GES2
100× RANDOM ∘ GES1
900× RANDOM ∘ GES2
Value-I.M. with seed hypervectors
GES1 GES2 TASK1 TASK2
Conditional bundle 1
COND1 ∘ TASK1 ∘ ρ(GES1)
Conditional bundle 2
COND2 ∘ TASK2 ∘ ρ(GES1)
CPT_{gest} Hypervector for $\Pr(Ges_t|Task_t, Ges_{t-1})$
Retrieving the value of $Pr(Ges_t = ges_1|Task_t = task_2, Ges_{t-1} = ges_1)$ from CPT_{gest}
CPT_{gest} ∘ TASK2 ∘ ρ(GES1) = COND2 + noise
dot(~COND2 , GES1) / (1000 × n) ≈ **0.1**

Ges_{t-1}	$Task_t$	Ges_t	$\Pr(Ges_t)$
ges_1	$task_1$	ges_1	0.8
ges_1	$task_1$	ges_2	0.2
ges_1	$task_2$	ges_1	0.1
ges_1	$task_2$	ges_2	0.9
...	...	...	...

Figure 2: Illustration of mapping a CPT to a hypervector and retrieving a probability from it.

2) Probabilistic reasoning layer. The second layer, highlighted in green, builds a high-level probabilistic world model of the application. In this work, we realize this model with a DBN that encodes the probabilistic relations among the tasks performed by the user and the observed hand-gestures and arm-positions, written as ges and pos, respectively. This DBN makes a first order Markov assumption for the temporal relations among tasks, gestures and positions. The probabilistic reasoning layer in our example framework performs filtering on this DBN. That is, it computes the posterior distribution over the task at time t, given all evidence over ges and pos to date. In practice, the inference algorithm maintains a belief over the task at time t and updates it at time $t+1$ given the evidence over $t+1$ and t in a recursive fashion (Russell and Norvig 2002). Specifically, for all values of $Task$ at time $t+1$:

$$\Pr(task_{t+1}|ges_{0:t+1}, pos_{0:t+1}) = \alpha \Pr(ges_{t+1}|task_{t+1}, ges_t) \cdot \Pr(pos_{t+1}|task_{t+1}, pos_t) \sum_{task_t} \Pr(task_{t+1}|Task_t) \cdot \Pr(Task_t|ges_{0:t}, pos_{0:t}), \tag{1}$$

where the variable $Task_t$ is marginalized as it remains unobserved; $\Pr(Task_t|ges_{0:t}, pos_{0:t})$ is calculated in the previous iteration with the evidence over ges and pos at time t and $t-1$; and α is a normalization constant that ensures that the posteriors sum to 1.

We map the DBN to HD space[2] by using the data structures discussed in the Hyperdimensional computing Section, as shown in Figure 2. First, we generate random hypervector seeds for each value of the $Task$ and Ges variables, and store them in the Value-I.M. Then, we randomly project the probabilities to HD space through scalar multiplication with new random hypervectors. Note that we scale the probabilities (by e.g. 10^3) such that we only have to store integers. Then, we bind this real-valued vector with the hypervector corresponding to the observed value (for example, GES1 in the first row of the CPT). As such, we can create bundles of real-valued hypervectors corresponding to each of the conditioned variables (e.g. we create a real-valued bundle for the Ges_t variable conditioned on $Task_t = task_1$ and $Ges_{t-1} = ges_1$, labeled COND1 in the figure). We can further superimpose these bundles by binding them to their corresponding conditioning values, as indicated by the labels *Conditional bundle*. Note that, in this example, we apply the permute operation to the values of Ges_{t-1} to explicitly indicate they were observed at $t-1$. Finally, we superimpose (through bundling) all these conditional probability vectors. This process of recursive binding and bundling allows us to store a single hypervector per CPT in the model.

At inference time, probabilities are recovered from the CPT hypervectors by unbinding them from their condition (normally observed or marginalized variables) and taking the normalized dot product with the hypervector corresponding to the value of interest (see histograms in the data structure section). For example, starting from $\text{CPT}_{\text{ges}_t}$ in Figure 2, the value of $\Pr(Ges_t = ges_1|Task_t = task_2, Ges_{t-1} = ges_1)$ can be recovered through $\text{dot}(\text{CPT}_{\text{ges}_t} \circ \text{TASK2} \circ \rho(\text{GES1}), \text{GES1})/(1000 \cdot n)$, where n is hypervector dimension and 1000 accounts for the scaling factor that facilitated integer storage during mapping. Inference can then be performed with Equation 1. As illustrated in Figure 2, unbinding the conditional variables hypervectors from the CPT hypervector produces a noisy version of COND2. Similarly, there can be information loss when recovering the final probabilities through the dot product. Thus, an important future work direction consists on deriving bounds on the error induced by parameter superposition and storage.

Despite the induced information loss, the motivation behind mapping probabilistic models to hypervectors is twofold: 1) The potential improvements in memory efficiency for large models, since the space complexity of the CPTs remains constant and depends only on the capacity[3] of the hypervectors (Frady, Kleyko, and Sommer 2018; Kleyko et al. 2018), unlike traditional tabular CPTs, which can have exponential space complexity.[4] We leave the precise memory

[2]This mapping method is inspired by (Cheung et al. 2019).

[3]Capacity refers to the limit on the number of reliably recoverable hypervectors from their superposition.

[4]Although there has been extensive research proposing constant space algorithms for DBNs (Darwiche 2001).

efficiency analysis for future work, but prior work has found that storing multiple models in superposition (Cheung et al. 2019) can lead to memory savings. For example, (Zhou, Muller, and Rabaey 2021) demonstrated parameter memory savings of one order of magnitude when implementing dual-stage classification using high-dimensional model superposition. The work in (Zeman, Osipov, and Bosnic 2021) also exploits superposition to achieve neural network compression rates of up to 100x in multi-task settings, compared to individual neural network deployment. Additionally, specialized HD computing hardware that generates the seed hypervectors dynamically at run-time (Kleyko, Frady, and Sommer 2021) have shown potential energy savings of between 5x and 10x (Menon et al. 2021) compared to a traditional machine learning implementations. 2) The second motivation for this mapping is that encoding the entire framework with the same primitives can be advantageous, as it can potentially enable reasoning and, eventually, learning across all levels of abstraction including the Control Layer described next. Moreover, a common representation can enable hardware re-use and integration across different types of machine learning workloads, a trend with growing interest in the hardware-algorithm co-optimization community (Galindez Olascoaga et al. 2019; Shafique et al. 2021).

3) Control layer. The last layer serves as an *interface* between the proposed framework and other external components of the system. There are several ways in which this layer can be realized, depending on the needs of the system and the application. This versatility is further facilitated by the used hypervector representation, as discussed through the two following examples:
- **Symbolic reasoning:** for example, to transfer knowledge to unknown contexts and environments through analogical reasoning. Section *Context Knowledge Transfer Using Symbolic Reasoning* discusses this idea in detail.
-**Reasoning to actuation (input-output) mapping:** the predicted user intent (i.e. belief in the set of tasks at time t given evidence) can, for example, be used to fine tune the output grip selection. One way of doing that in HD space is by augmenting the functionality of HD computing implementations of state-automata-based control (Osipov, Kleyko, and Legalov 2017) or recall of reactive behaviour (Neubert, Schubert, and Protzel 2017). In the recall technique, illustrated in Figure 1, a composite hypervector representation encodes sensor-action pairs, such that behavior can be resembled at run-time by extracting (through unbinding) the action corresponding to the current sensor value. We posit that the posterior distribution from the layer below could further be used to weight this recall representation, an interesting direction that we leave for future work.

Experiments

Experimental setup and dataset

We validated our approach with the dataset in (Zhou, Muller, and Rabaey 2021), which includes measurements from a 64-channel EMG electrode array with an on-board tri-axial accelerometer, shown in Figure 3.a. We consider 3 hand-gestures and 4 arm positions, as illustrated in 3.b and 3.c.

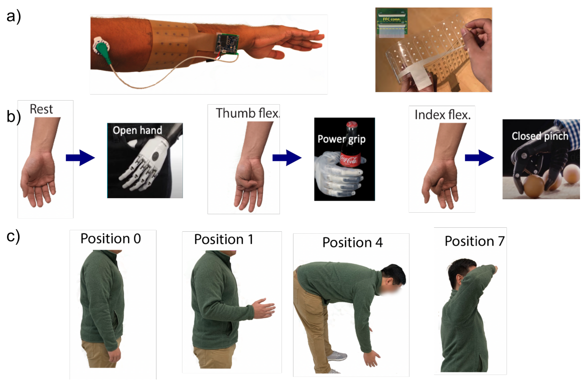


Figure 3: Experimental setup, based on (Zhou, Muller, and Rabaey 2021; Moin et al. 2021).

In addition, we define 3 tasks based on common activities of daily living and based on the available gestures/grips and postures from the aforementioned dataset: brushing hair, tying shoelace and opening jar. Sequential data over the different tasks is not available in (Zhou, Muller, and Rabaey 2021), therefore, we generate a 10000 sample sequential dataset by sampling an ideal DBN modeled with expert knowledge and we use a random 70%, 30% train-test split over 10 trials to evaluate the performance of the two lowest layers of the proposed framework.

Task description

The task we evaluated consisted on predicting the most likely $Task$ at time t, given evidence over Ges and Pos over time. This is done through the filtering algorithm described in Equation 1. Note that the evidence variables in the DBN, Ges and Pos, are the output of the classification of EMG and accelerometer signals in the perception layer. In particular, EMG signals are classified as hand-gestures through an HD computing implementation of a nearest centroid classifier, and accelerometer signals are classified as arm-positions through an SVM classifier.

Moreover, we evaluate two DBNs with different structures: the *complex* model has the same structure as the model depicted in Figure 1, and the *simple* model ignores the time relationships among gestures and positions. That is, at any time-slice t of the DBN, ges_t and pos_t only depend on $task_t$. We evaluate these two model structures to compare how their hypervector CPT implementation degrades in comparison to the traditional tabular CPT implementation.

Results

Figure 4 shows an example of the posterior distributions evaluated with the aforementioned DBNs (simple in blue and complex in red) when using a standard tabular CPT representation (cross markers) and when mapping CPTs to hypervectors (circle markers). The real distribution (from the DBN we sampled to generate the datasets) is shown with a black dotted line. It is clear that the posterior calculated by the complex model is closer to the real distribution. However, note that the hypervector CPT mapping of the com-

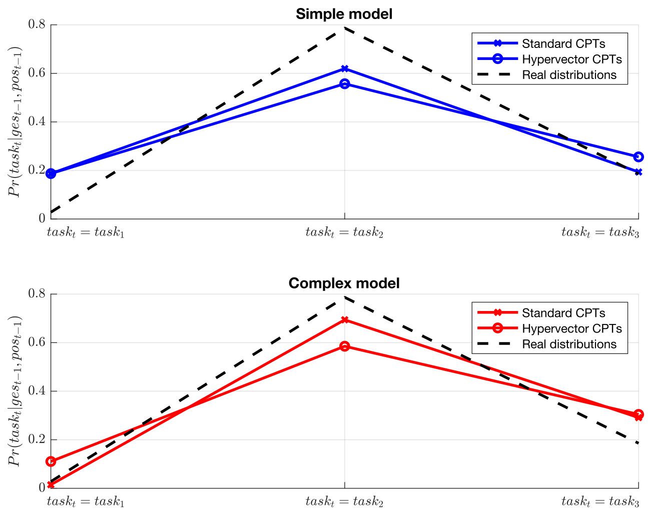


Figure 4: Examples of posterior distributions.

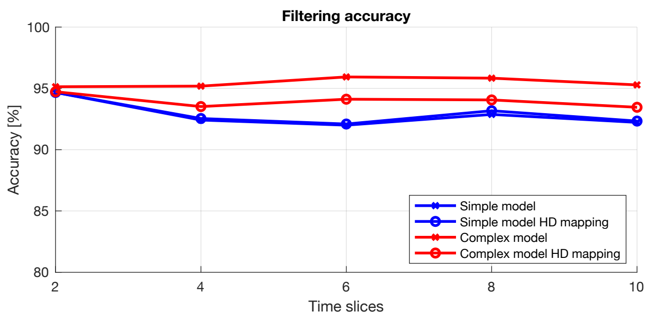


Figure 5: Filtering accuracy comparison over 10 time slices.

plex model degrades more than the simple model's. This is expected due to the limited superposition capacity of the hypervector representation, which is strained more by the complex model due to the additional variable-wise dependencies among Ges and Pos.

Figure 5 compares prediction accuracy for all models. Overall, filtering accuracy is preserved throughout the 10 time slices, and the complex model shows superior performance than the simple one. Yet, once again, the hypervector representation of the complex model loses nearly 3% accuracy, whereas the hypervector representation of the simple one does not lead to significant accuracy degradation.

Overall, there is a clear trade-off between hypervector capacity (driven by its representation type and dimension), model complexity, and memory efficiency, which we leave for future work. An interesting direction would be to systematically quantify the superposition capacity of hypervectors when used to represent graphical probabilistic models such as the DBN studied in this work.

Context Knowledge Transfer Using Analogical Reasoning

The Control Layer can allow us to deduce generalizable outcomes that can be carried from one context to another. This is especially important because learning is typically performed based on sporadic samples that are attributed to a limited number of contexts. Some samples may exhibit common features with respect to learnt tasks among different contexts, whereas other samples lead to different task definitions depending on the context. The question is: how can we optimize our learning method such that it can systematically transfer task definitions between contexts so that our system can adapt to unknown settings in new contexts? This section presents some examples of how the symbolic and, more specifically, analogical reasoning capabilities of HD computing can be exploited for the transfer of knowledge across context in the prosthetic use case.

For example, suppose that the DBN in Figure 1 was trained through the interactions of a prosthetic arm with objects in a kitchen, and that the DBN's current task prediction (i.e. $Task_t$) is opening jar. We can create a composite hypervector that describes this situation, defined in terms of the current context (location), and task (activity):

$$\mathsf{H}_1 = \mathsf{LOC} \circ \mathsf{KIT} + \mathsf{ACT} \circ \mathsf{OJ},$$

where $\mathsf{LOC} \circ \mathsf{KIT}$ denotes that the location in "situation 1" is the kitchen and $\mathsf{ACT} \circ \mathsf{OJ}$ denotes that the current activity is opening jar.

Because our DBN was trained solely with data from a kitchen context, it may be unable to generalize towards the correct prediction of activities in a new context, such as a "front-door" location. Instead of seeking to gather more training data and re-train the DBN, we can explicitly encode our expert knowledge about all the different scenarios in the Control Layer, such that the framework can automatically exploit this information at run-time.

For example, we know that opening a door in the "front-door" context requires a twisting motion similar to the one used to open a jar in the kitchen context. Thus, we can encode the "front-door" twisting situation as the hypervector:

$$\mathsf{H}_2 = \mathsf{LOC} \circ \mathsf{FDR} + \mathsf{ACT} \circ \mathsf{OD},$$

where $\mathsf{LOC} \circ \mathsf{FDR}$ denotes that "situation 2" takes place at a front-door location and $\mathsf{ACT} \circ \mathsf{OD}$ denotes that the twisting activity in this situation is opening the door.

Then, we can instruct our framework to reason analogically through substitution about the two situations, with a query similar to "What is the Dollar of Mexico?" (Kanerva 2010). Specifically, in our case, we wish the Control Layer to answer the question, "what is the analogous of opening a jar when the user is in a front-door environment?".

First, hypervectors corresponding to the two situations must be bound, such that the values occupying the same keys are bound (i.e. kitchen is to front-door as open-jar is to open-door):

$$\begin{aligned}\mathsf{H}_1 \circ \mathsf{H}_2 &= \mathsf{LOC} \circ \mathsf{KIT} \circ \mathsf{LOC} \circ \mathsf{FDR} + \mathsf{LOC} \circ \mathsf{KIT} \circ \mathsf{ACT} \circ \mathsf{OD} \\ &+ \mathsf{ACT} \circ \mathsf{OJ} \circ \mathsf{LOC} \circ \mathsf{FDR} + \mathsf{ACT} \circ \mathsf{OJ} \circ \mathsf{ACT} \circ \mathsf{OD} \\ &= \mathsf{KIT} \circ \mathsf{FDR} + \mathsf{OJ} \circ \mathsf{OD} + noise \\ &\approx \mathsf{KIT} \circ \mathsf{FDR} + \mathsf{OJ} \circ \mathsf{OD}.\end{aligned}$$

Then, the desired analogical query can simply be answered by probing the bound structure above with the open-jar (OJ)

hypervector:

$$\begin{aligned}\mathsf{H}_1 \circ \mathsf{H}_2 \circ \mathsf{OJ} &= \mathsf{KIT} \circ \mathsf{FDR} \circ \mathsf{OJ} + \mathsf{OJ} \circ \mathsf{OD} \circ \mathsf{OJ} \\ &= noise + \mathsf{OD} \approx \mathsf{OD}.\end{aligned}$$

Our framework is thus capable of concluding that the analogous to opening jar in a new "front door" environment is opening a door. Note that no training data from the new front-door environment nor re-tuning of the DBN were required to reach to this conclusion. This is useful when expert knowledge is available and training data is costly.

These symbolic reasoning mechanisms could further be used to provide feedback to the probabilistic model for continual learning and adaptation, an interesting direction for future work.

Related work

This work presents a hybrid framework comprising different types of reasoning and computational paradigms. As such, the body of related work is vast and spans multiple fields. For brevity, we focus mostly on related neurosymbolic computing strategies and existing hybrid HD computing/VSA solutions, but we are aware of many other relevant works, which we would like to further discuss in future and extended versions of this paper.

Neurosymbolic approaches. Several state-of-the-art approaches in complex event processing have attempted to leverage expert knowledge by diving the problem into low-level perception and high-level reasoning, and enabling neural networks to learn the former from data and emulate the latter from user defined rules (Xing et al. 2020). Yet, these approaches can lack expressivity at the symbolic level and can suffer from explainability issues. Other approaches have addressed these limitations by still leveraging neural networks to process sub-symbolic (e.g. sensor) data, but relying on (probabilistic) logic languages to inject expert knowledge on higher level information, such as the rules of complex events (Vilamala et al. 2021; Apriceno, Passerini, and Serafini 2021).

Related HD computing/VSA approaches. Recent works explore how VSAs and Holographic Reduced Representations can be used to propose novel neural network architectures capable of inferring heterogeneous knowledge, such as pattern recognition together with spatial information (Weiss, Cheung, and Olshausen 2016; Yılmaz 2015). Analogical reasoning has been one of the tools used for such hybrid applications. For a detailed overview of HD computing applications, we refer the reader to (Kleyko et al. 2021c).

Aspects of our framework that differ from the work highlighted above. The framework we propose is conceptually modular and we believe it has the potential of leveraging suitable machine learning and AI strategies at each layer. Thus, a potential future work direction would be to re-implement the perception and probabilistic reasoning layers through the approaches highlighted above. An important difference with end-to-end hybrid and neurosymbolic implementations is the role of symbolic reasoning at the Control Layer in our framework. Unlike the techniques mentioned above, which exploit logical statements to improve the performance on the task of interest[5], we plan to use symbolic reasoning to interface with elements outside the framework (actuation control), and to provide feedback to lower levels towards dynamic and context dependent adaptive and continual learning. This is further discussed in the following section.

Discussion

Implications of the proposed framework

- Note that the framework we proposed assumes that each layer is trained, learned or stated (i.e. symbolic knowledge) independently through the most appropriate technique. The believe that the HD-computing-enabled integration discussed in this work can be best exploited at inference-time or run-time.
- The proposed probabilistic model within our framework is rather simple since it was trained with synthetic data sampled from an ideal model constructed with expert knowledge. As such, we were able to store it in superposition without significant performance degradation. For real life applications, it will be necessary to perform more complete analyses on the capacity of the models and related trade-offs.
- Overall, the considered use case benefits from the three layers in the proposed framework. This opens the door for further investigation into the adaptation of this hierarchical framework for different use cases, which may benefit from a different configuration of our framework in terms of, the classifier in the perception layer, the probabilistic model and inference algorithm in the middle layer, or the type of symbolic reasoning or actuation control for the top layer.

Future work

- An interesting direction would be to explore the use of tractable probabilistic models, such as Probabilistic Circuits (Choi, Vergari, and den Broeck 2020) for the probabilistic inference layer. This would involve evaluating whether such a model could be mapped to the HD space, similar to the DBN example discussed herewith.
- Beyond analogical reasoning, this framework can benefit from other forms of symbolic and logical reasoning and the various state-of-the-art techniques that enable its efficient execution.
- As mentioned in the Analogical Reasoning section, we believe the control layer could coordinate a process of active learning. Thus we need to review whether existing strategies in the fields of transfer learning, meta learning or reinforcement learning could be appropriate for this, or whether a different technique may be required.
- Another future work direction would be to derive bounds on the error propagation caused by the hypervector representations, and especially when storing complex probabilistic models with high-dimensional superposition.
- Finally, an important future work direction consists on comparing the performance of the proposed framework with other hybrid strategies, such as neurosymbolic computing.

[5]Although, as mentioned, we believe these strategies could also be leveraged by our framework.

Acknowledgements

We thank the reviewers of this paper for their helpful comments and Denis Kleyko for his feedback on the camera-ready version. We also thank Andy Zhou for sharing the datasets used in this work and Jennifer Ruffing for useful discussions. This work was supported in part by the VIP and HYDDENN programs sponsored by DARPA. This work was also supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. 1752814. Support was also received from sponsors of Berkeley Wireless Research Center. Any opinions, findings, and conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, DARPA, or the U.S. government.

References

Apriceno, G.; Passerini, A.; and Serafini, L. 2021. A Neuro-Symbolic Approach to Structured Event Recognition. In *TIME*.

Cheung, B.; Terekhov, A.; Chen, Y.; Agrawal, P.; and Olshausen, B. 2019. Superposition of many models into one. *Advances in Neural Information Processing Systems*, 32: 10868–10877.

Choi, Y.; Vergari, A.; and den Broeck, G. V. 2020. Probabilistic Circuits: A Unifying Framework for Tractable Probabilistic Models.

Darwiche, A. 2001. Constant-space reasoning in dynamic Bayesian networks. *Int. J. Approx. Reason.*, 26: 161–178.

Diao, C.; Kleyko, D.; Rabaey, J. M.; and Olshausen, B. A. 2021. Generalized Learning Vector Quantization for Classification in Randomized Neural Networks and Hyperdimensional Computing. In *International Joint Conference on Neural Networks (IJCNN)*, 1–9.

Eliasmith, C.; Stewart, T. C.; Choo, X.; Bekolay, T.; DeWolf, T.; Tang, Y.; and Rasmussen, D. 2012. A Large-scale Model of the Functioning Brain. *Science*, 338(6111): 1202–1205.

Espinosa, M.; and Nathan-Roberts, D. 2019. Understanding Prosthetic Abandonment. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 63: 1644 – 1648.

Frady, E. P.; Kleyko, D.; and Sommer, F. T. 2018. A Theory of Sequence Indexing and Working Memory in Recurrent Neural Networks. *Neural Computation*, 30: 1449–1513.

Galindez Olascoaga, L. I.; Meert, W.; Shah, N.; Verhelst, M.; and Van den Broeck, G. 2019. Towards hardware-aware tractable learning of probabilistic models. *Advances in Neural Information Processing Systems 32 (NeurIPS)*, 32.

Gayler, R. W. 1998. Multiplicative binding, representation operators and analogy. *Advances in analogy research: integration of theory and data from the cognitive, computational and neural sciences*.

Gayler, R. W. 2003. Vector Symbolic Architectures Answer Jackendoff's Challenges for Cognitive Neuroscience. In , ed., *Joint International Conference on Cognitive Science (ICCS/ASCS)*, 133–138. .

Jeon, H.-A. 2014. Hierarchical processing in the prefrontal cortex in a variety of cognitive domains. *Frontiers in Systems Neuroscience*, 8.

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

Kanerva, P. 2010. What we mean when we say" What's the dollar of Mexico?": Prototypes and mapping in concept space. In *2010 AAAI fall symposium series*.

Kanerva, P.; et al. 1997. Fully distributed representation. *PAT*, 1(5): 10000.

Kleyko, D.; Davies, M.; Frady, E. P.; Kanerva, P.; Kent, S. J.; Olshausen, B. A.; Osipov, E.; Rabaey, J. M.; Rachkovskij, D. A.; Rahimi, A.; et al. 2021a. Vector Symbolic Architectures as a Computing Framework for Nanoscale Hardware. *arXiv preprint arXiv:2106.05268*.

Kleyko, D.; Frady, E. P.; and Sommer, F. T. 2021. Cellular Automata Can Reduce Memory Requirements of Collective-State Computing. *IEEE Transactions on Neural Networks and Learning Systems*, 99(PP): 1–13.

Kleyko, D.; Rachkovskij, D. A.; Osipov, E.; and Rahimi, A. 2021b. A Survey on Hyperdimensional Computing aka Vector Symbolic Architectures, Part I: Models and Data Transformations. *arXiv:2111.06077*, 1–27.

Kleyko, D.; Rachkovskij, D. A.; Osipov, E.; and Rahimi, A. 2021c. A Survey on Hyperdimensional Computing aka Vector Symbolic Architectures, Part II: Applications, Cognitive Models, and Challenges . *arXiv*, 1–36.

Kleyko, D.; Rahimi, A.; Rachkovskij, D. A.; Osipov, E.; and Rabaey, J. M. 2018. Classification and Recall With Binary Hyperdimensional Computing: Tradeoffs in Choice of Density and Mapping Characteristics. *IEEE Transactions on Neural Networks and Learning Systems*, 29: 5880–5898.

Menon, A.; Sun, D.; Aristio, M.; Liew, H.; Lee, K.; and Rabaey, J. M. 2021. A Highly Energy-Efficient Hyperdimensional Computing Processor for Wearable Multi-modal Classification. *IEEE Biomedical Circuits and Systems Conference (BioCAS)*, 1–4.

Mitrokhin, A.; Sutor, P.; Fermuller, C.; and Aloimonos, Y. 2019. Learning Sensorimotor Control with Neuromorphic Sensors: Toward Hyperdimensional Active Perception. *Science Robotics*, 4(30): 1–10.

Moin, A.; Zhou, A.; Rahimi, A.; Menon, A.; Benatti, S.; Alexandrov, G.; Tamakloe, S.; Ting, J.; Yamamoto, N.; Khan, Y.; et al. 2021. A wearable biosensing system with in-sensor adaptive machine learning for hand gesture recognition. *Nature Electronics*, 4(1): 54–63.

Neubert, P.; Schubert, S.; and Protzel, P. 2017. *Learning vector symbolic architectures for reactive robot behaviours*. Universitätsbibliothek Chemnitz.

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

Osipov, E.; Kleyko, D.; and Legalov, A. 2017. Associative Synthesis of Finite State Automata Model of a Controlled

Object with Hyperdimensional Computing. In *Annual Conference of the IEEE Industrial Electronics Society (IECON)*, 3276–3281.

Park, C.; Took, C. C.; and Seong, J.-K. 2018. Machine learning in biomedical engineering. *Biomedical Engineering Letters*, 8.

Plate, T. 1994a. Distributed representations and nested compositional structure (PhD thesis). *University of Toronto, Toronto, ON, Canada*.

Plate, T. A. 1994b. Estimating Analogical Similarity by Dot-products of Holographic Reduced Representations. In *Advances in Neural Information Processing Systems (NIPS)*, 1109–1116.

Rachkovskij, D. A. 2001. Representation and Processing of Structures with Binary Sparse Distributed Codes. *IEEE Transactions on Knowledge and Data Engineering*, 3(2): 261–276.

Rachkovskij, D. A.; Kussul, E. M.; and Baidyk, T. N. 2013. Building a World Model with Structure-sensitive Sparse Binary Distributed Representations. *Biologically Inspired Cognitive Architectures*, 3: 64–86.

Rachkovskij, D. A.; and Slipchenko, S. V. 2012. Similarity-based Retrieval with Structure-Sensitive Sparse Binary Distributed Representations. *Computational Intelligence*, 28(1): 106–129.

Rahimi, A.; Kanerva, P.; Benini, L.; and Rabaey, J. M. 2018. Efficient biosignal processing using hyperdimensional computing: Network templates for combined learning and classification of exg signals. *Proceedings of the IEEE*, 107(1): 123–143.

Russell, S.; and Norvig, P. 2002. Artificial intelligence: a modern approach.

Shafique, M.; Theocharides, T.; Reddy, V. J.; and Murmann, B. 2021. TinyML: Current Progress, Research Challenges, and Future Roadmap. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, 1303–1306. IEEE.

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

Vilamala, M. R.; Xing, T.; Taylor, H.; Garcia, L.; Srivastava, M. B.; Kaplan, L. M.; Preece, A. D.; Kimmig, A.; and Cerutti, F. 2021. Using DeepProbLog to perform Complex Event Processing on an Audio Stream. *Tenth International Workshop on Statistical Relational AI*.

Weiss, E.; Cheung, B.; and Olshausen, B. 2016. A neural architecture for representing and reasoning about spatial relationships.

Xing, T.; Garcia, L.; Vilamala, M. R.; Cerutti, F.; Kaplan, L. M.; Preece, A. D.; and Srivastava, M. B. 2020. Neuroplex: learning to detect complex events in sensor networks through knowledge injection. *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*.

Yılmaz, Ö. 2015. Symbolic Computation Using Cellular Automata-Based Hyperdimensional Computing. *Neural Computation*, 27: 2661–2692.

Zeman, M.; Osipov, E.; and Bosnic, Z. 2021. Compressed Superposition of Neural Networks for Deep Learning in Edge Computing. In *International Joint Conference on Neural Networks (IJCNN)*, 1–8.

Zhou, A.; Muller, R.; and Rabaey, J. M. 2021. Memory-Efficient, Limb Position-Aware Hand Gesture Recognition using Hyperdimensional Computing. In *tinyML Research Symposium (tinyML)*, 1–8.

Zhou, Z.; Chen, X.; Li, E.; Zeng, L.; Luo, K.; and Zhang, J. 2019. Edge Intelligence: Paving the Last Mile of Artificial Intelligence with Edge Computing. *Proceedings of the IEEE*, 107(8): 1738–1762.