Theme: 2nd Workshop on Metacognitive Prediction of AI Behavior

Hyperdimensional Computing for Metacognition

Peter Sutor, *University of Maryland, College Park, MD, 20770, USA*
Cornelia Fermüller, *University of Maryland, College Park, MD, 20770, USA*
Yiannis Aloimonos, *University of Maryland, College Park, MD, 20770, USA*

Abstract—Hyperdimensional Computing (HDC) is a computing paradigm that operates in closed algebras of high dimensional vectors, called hypervectors. HDC's usage affords many interesting properties in both symbolic level AI and Machine Learning. Namely, the ability to rapidly learn policies that approximate models in its high dimensional space. In this paper, we discuss how HDC can be used as a metacognitive framework in which such meta-models can be learned and merged together. Moreover, we present methodologies derived from prior work in HDC that further allow desired properties of metacognition to manifest; such as explainability of black-box models, self adaption and online learning, and perceptual grounding.

Metacognition in AI is the idea of reasoning about the AI system itself. Hyperdimensional Computing (HDC) is a computing paradigm in which an AI can be observed and modeled through algebraic statements. Clearly, it is quite natural to employ HDC to facilitate metacognition. In this paper, we discuss prior results in HDC recontextualized for metacognition, in order to demonstrate how HDC can be used to facilitate metacognition for any AI system, be they symbolic, neurosymbolic, or machine learning derived. First, we describe HDC and some key operations and properties. Then, we focus on several works, referred to as "Case Studies" that showcase metacognition via HDC. Finally, we discuss other methodologies in HDC that can further facilitate the gain of various metacognitive capabilities.

Hyperdimensional Computing

At its core, Hyperdimensional Computing is the realization that most computing occurs in minimal representations of information necessary for the computation (integers, floating points, true/false, etc.), which is incongruous with computation in real life brains; these are inherently high dimensional in nature, far beyond these minimal representations. Instead, the brain must operate on instances of high dimensional representations. To model these operations, an algebra of hypervectors must be constructed to approximate the empirical behavior, as noted by Pentti Kanerva [1].

We will focus on the simplest, but most ubiquitous form of HDC; random binary strings as hypervectors. These are immediately mappable to computer operation and circuits, and serve as digital approximation of computational behavior in high dimensional strings of information. Consider the space of vectors $\mathbf{V} \in \{0,1\}^n$, or binary strings of length n. We consider two algebras. The first employs a *binding* operator (our abstraction of a product in this algebra) of bit-wise exclusive-or (XOR) of 2 vectors, $\mathbf{A} \oplus \mathbf{B}$. This binds two pieces of information together. Namely, as XOR is a difference operator, it produces a digital involution between $\mathbf{A}$ and $\mathbf{B}$. In other words, an invertible boolean function mapping the two to each other. Simply XOR with the same vector to remove it from the mapping. The *bundling* operator (the abstraction of superposition in this algebra) is defined as the *Consensus Sum*, or "best vote" across the bits of each component of the hypervector terms being superposed, producing a logical vector:

$$\mathbf{V_1} + \mathbf{V_2} + \cdots + \mathbf{V_m} = \left(\sum_{i=1}^{m} \mathbf{V_i} > m - \sum_{i=1}^{m} \mathbf{V_i} \right) \quad (1)$$

If component j had more 1's than 0's, that component will have a 1 as a result and 0 otherwise. For an even number of terms, a random tie-breaker term V_{m+1} can

Digital Object Identifier 10.1109/XXX.0000.0000000

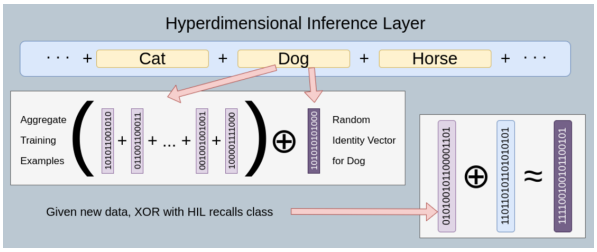

FIGURE 1. The Hyperdimensional Inference Layer (HIL). The inferred class has the lowest Hamming Distance from its hypervector to the XOR of an input hypervector and the HIL.

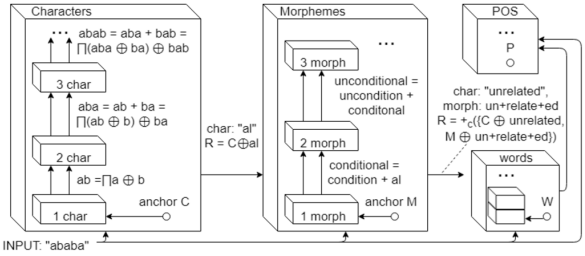

FIGURE 2. Process for how semantics can be learned unsupervised and then be supervised via HDC.

be added to the sum. If you bundle a set of bindings, you compute a model that digitally maps inputs vector to output vectors. If the input vectors are perceptual in nature, and output vectors are output signals in a brain, you compute a digital function that associates (and classifies) perception as actions/activations of the brain. This gives rise to what will be referred to as the Hyperdimensional Inference Layer (HIL), as shown in Figure 1, the common workhorse of HDC. We will also consider a second algebra, which can be useful for sequences, where the binding operator is repeated permutation of bits under a certain permutation pattern. Given permutation pattern $\mathbf{P}$ (a vector of new component indices), you can bind a hypervector A to symbolic position k in a sequence by:

$$\mathbf{A}[\mathbf{P}^k] \tag{2}$$

Note that $\mathbf{P}^k$ simply calls (2) with $\mathbf{A} \leftarrow \mathbf{P}$ and $k \leftarrow k-1$. By computing the inverting mapping $\mathbf{P}^{-1}$ such that $\mathbf{P}[\mathbf{P}^{-1}]$ is simply the identity mapping, any binding under permutation can be shifted at will.

As the length n of the hypervector approaches hyperdimensional values, typically $n \approx 2^{13}$, or more, the space of possible vectors rapidly grows. Random correlations become unlikely. Thus, nearest matches become significant so long as the dip in Hamming Distance of the match is statistically significant. These can range from several standard deviations to dozens in the Binomial Distribution of n coin flips. Ideally, hypervectors are random, unless they come from perception, such as in [2]. Perception-based hypervectors are computed from features, and this encoding process is an art that is developed for each problem. Encodings of inputs for each output class should characterize their differences, while being as similar to in-class training examples as possible. Due to the recent progression of Deep Learning, neural networks can be directly converted to hypervector models [3], [4]. Still, finding ideal encodings remains an art.

Case Studies

In this section, we cover case studies that show how HDC performs metacognition to learn metamodels.

Life-Long Learning of Semantics [5]

In this work, Life-Long Learning of HD models is explored. Semantics can be learned for unsupervised data by converting it to symbolic hypervectors and binding positional information to it by sequences. This employs both algebras that bind with XOR and bind with sequences as permutations. A spring-mass model for tension is developed involving hypervectors as locations of the masses, occurrence counts the mass, and co-occurrence counts as springs connecting hypervectors in a knowledge graph. Using physics derived equations and simulated annealing, random hypervector are relaxed into low-energy states of low tension, thereby finding appropriate hypervectors to reflect the knowledge graph. Oracles can be employed on top of this to add supervision labels and build on this knowledge graph, an example of which is shown on textual data in Figure 2. From a metacognitive standpoint, this shows that HDC doesn't require an explicit AI to meta-model; unsupervised data and even raw physics can be used as a model to employ metacognition on. Furthermore, you can build on any metacognitive HD model through arbitrary supervision from oracles (other models) to build a self-consistent meta-model.

Hyperdimensional Active Perception [2]

In this work, we demonstrate that HDC can perform Machine Learning directly on perceptual data to solve tasks. From a computational standpoint, it makes no difference if an actual model exists as a black-box mapping input to output. HDC can infer what the model should be under its own algebra by an HIL such as Figure 1. Given perception describing ego-motion, the

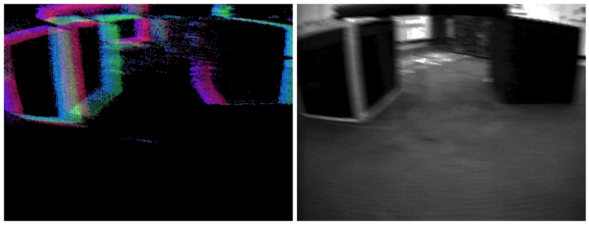

FIGURE 3. Comparison of time slices of Event Cameras (left) versus the actual gray-scale image (right). Note how sparse event data is.

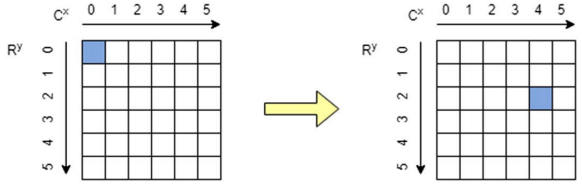

FIGURE 4. By permutation binding, pixel intensities can be moved to their respective position in an image via repeating 2 permutation patterns **R** and **C**, representing the row and column index shifts from the origin.

ego-motion itself can be predicted by raw algebraic bindings of perception hypervectors to output hypervectors tied to, in this case study, velocity bins in relevant dimensions for navigation. The perception comes from a specialized camera known as an Event Camera, which only produces visual data by asynchronously recording time stamps of pixels that undergo significant intensity changes. Therefore, motion directly causes sparse visual data to occur. These are binned in linear projections through time to create sparse 2D images, such as those in Figure 3, then encoded to hypervectors by permuting hypervectors representing pixel intensities to specific positions, then superposing the pixels, employing the second algebra described earlier by (2). By maintaining two permutation patterns, one for row and one for column location, an image becomes a nested subsequence of pixel intensities that have been shifted as in Figure 4. Velocities experienced over time are associated to time-slices and grouped to corresponding velocity bins. These velocities can then be predicted sequentially to relive “memories" of how to navigate an environment. As hypervector models are sparse learners, robust 2 kilobyte models are learned within a fraction of a second in a CPU on mere seconds of data, where traditional methods suchs as CNNs require hours of data and training time, GPUs, and gigabytes of memory to achieve that performance.

Symbolic Representation and Learning [3]

In this work, prexisting hashing networks are employed as black-box models that produce short, rankable hashcodes on the order of 100 bits or less. Hyperdimensional Models are learned on these hashcodes by projecting them to hyperdimensional lengths via repetition and random permutation. Three such vastly different hashing networks of increasing performance are then fused together by giving each network its own random hypervector identifying it, then binding its model to it by XOR, and superpositioning the models together via 1, as shown in Figure 5 Interestingly, this boosts the performance overall by an unexpectedly high amount, since the hypervector models give more voting power in the consensus to models that are a good fit, leveraging the pros and cons of each modality. This work shows several things in the domain of metacognition. Firstly, hypervector models can indirectly learn black-box models by only interacting with their outputs, thus learning a meta-model that approximates the original in a different space. Secondly, these meta-models directly transfer their knowledge in the HDC algebra despite being different modalities.

HD-Glue [4]

In this final case study, many of the previous ideas are tied together in the context of neural network architectures. This work shows that the results of [3] can be extrapolated to deep neural networks. By taking output signals, i.e., embeddings, of neural networks just before classification, all representational power can be converted to hypervector models by approximating the embeddings with hypervectors. This process is shown in Figure 6. After conversion, the results show similar consensus properties such as those in [3], particularly when architectures are very different. Furthermore, a method of error correction is developed which can help hypervectors better themselves by finding models for parts of the dataset they got wrong. When these are glued together into one model, a consensus with error correction can be made to improve performance. From a metacognitive standpoint, this shows a clear meta-model learning throughline where an understanding of a neural network model can be gleaned through easily explainable policies in the form of hypervector algebras. That this can be done on arbitrary architectures is of particular interest to the metacognition community.

HDC for Metacognition

In conclusion, we note other features in HDC:

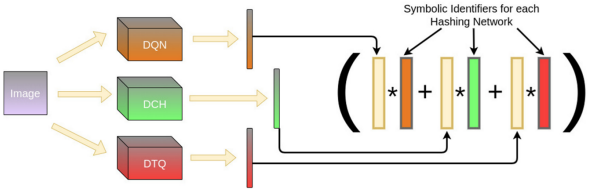

FIGURE 5. Hashcodes from hashing networks are approximated by hypervector models and then fused together into a single into a consensus architecture to boost performance by assigning random hypervector identifiers to each model.

Topos Theory

Deep Learning is often constrained by requiring smooth gradients in continuous spaces. Indeed, we find neural networks often perform best when non-linearities such as ReLU are added to deal with these constraints. From a Category Theoretical perspective, almost all AI is confined by Topos Theory, a generalization of topology in category theory. Recent work in [6] shows that HDC has the potential to circumvent this limitation by leveraging its symbolic representations as direct morphisms that stitch together disparate topological spaces as subspaces in Hamming Spaces. In this paper, Hamming Balls are used to achieve this, which are subspaces of all vectors a certain Hamming Distance away from a centroid hypervector. As such, metacognitive models can be embedded within these subspaces so long as the hypervectors are long enough to make large enough spaces.

Non-overfitting Perfect Models

Hamming Balls can also be employed to clip an existing AI to 100% accurate predictions in the HDC space. By going through the training set and finding the largest hamming ball where the HIL is always correct, a metamodel is made that can get 100% correct accuracy on the training set but not overfit on the test set. This reveals where a model fails, and more importantly, why, via a metacognitive HD model.

Hypervector Memories

Typically, HD models ignore the order data is presented in. Effectively, re-ordering doesn't produce a different HD model. However, this isn't a requirement. Order can be imposed by permutation or binding symbolic hypervectors. Multiple hypervectors can be superposed to form local memories. This was implicit in the ego-motion task in [2], but can be made explicit for online learning of metacognitive models over time.

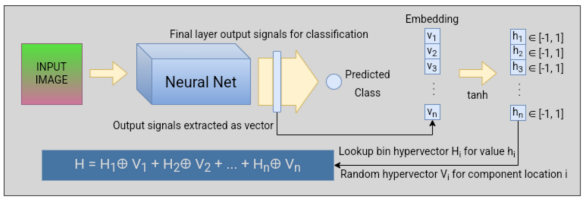

FIGURE 6. Embedding to hypervector conversion.

Finite State Automata (FSA)

As HDC is capable of converting from hypervector form to symbolic form at will, it can represent nearly any mathematical structure in vector encodings. This, of course, includes any Finite State Automata. Any Metacognitive model that operates as a FSA, graph, etc., can exist in either form, granting native direct integration and explainability capabilities.

REFERENCES

1. P. Kanerva, "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors," *Cognitive computation*, vol. 1, pp. 139–159, 2009. (Journal)
2. M. Anton, P. Sutor, C. Fermüller, and Y. Aloiomonos, "Learning sensorimotor control with neuromorphic sensors: Toward hyperdimensional active perception," *Science Robotics*, vol. 4, no. 30, 2019. (Journal)
3. M. Anton, P. Sutor, D. Summers-Stay, C. Fermüller, and Y. Aloiomonos, "Symbolic representation and learning with hyperdimensional computing," *Frontiers in Robotics and AI*, vol. 7, pp. 63, 2020. (Journal)
4. P. Sutor, D. Yuan, D. Summers-Stay, C. Fermüller, and Y. Aloiomonos "Gluing neural networks symbolically through hyperdimensional computing, " *Proc. 26th International Joint Conference on Neural Networks (IJCNN)*, 2022, pp. 1-10. (Conference proceedings)
5. P. Sutor, D. Summers-Stay, and Y. Aloiomonos, "A computational theory for life-long learning of semantics," *Artificial General Intelligence: 11th International Conference*, pp. 217-226, 2018. (Conference proceedings)
6. R. Faraone, P. Sutor, C. Fermüller, and Y. Aloimonos, "Vector Symbolic Sub-objects Classifiers as Manifold Analogues," *Proc. 27th International Joint Conference on Neural Networks (IJCNN)*, pp. 1-10. (Conference proceedings)