2023 57th Annual Conference on Information Sciences and Systems (CISS) | 978-1-6654-5181-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/CISS56502.2023.10089705

Towards an Improved Hyperdimensional Classifier for Event-Based Data

Neal Anwar, Chethan Parameshwara, Cornelia Fermüller, Yiannis Aloimonos

Abstract— **Hyperdimensional Computing (HDC) is an emerging neuroscience-inspired framework wherein data of various modalities can be represented uniformly in high-dimensional space as long, redundant holographic vectors. When equipped with the proper Vector Symbolic Architecture (VSA) and applied to neuromorphic hardware, HDC-based networks have been demonstrated to be capable of solving complex visual tasks with substantial energy efficiency gains and increased robustness to noise when compared to standard Artificial Neural Networks (ANNs). HDC has shown potential to be used with great efficacy for learning based on spatiotemporal data from neuromorphic sensors such as the Dynamic Vision Sensor (DVS), but prior work has been limited in this arena due to the complexity and unconventional nature of this type of data as well as difficulty choosing the appropriate VSA to hypervectorize spatiotemporal information. We present a bipolar HD encoding mechanism designed for encoding spatiotemporal data, which captures the contours of DVS-generated time surfaces created by moving objects by fitting to them local surfaces which are individually encoded into HD vectors and bundled into descriptive high-dimensional representations. We conclude with a sketch of the structure and training/inference pipelines associated with an HD classifier, predicated on our proposed HD encoding scheme, trained for the complex real-world task of pose estimation from event camera data.**

I. Introduction

The increasing volumes of rich data generated by the sensors edge devices such as mobile phones and other appliances under the purview of the Internet of Things (IoT) present a desirable avenue for the application of machine learning algorithms to solve complex cognitive tasks. However, the strict and uncompromising computational and memory limitations typical to these edge devices make the prospect of running the high-complexity machine learning processes generally used on non-edge devices equipped with significantly greater processing power unfeasible; it is possible to send edge-borne data to the cloud for processing, but this solution suffers from lack of scalability and response delay. An alternative method of learning capable of running on less powerful edge devices without sacrificing accuracy, robustness, or generalizability is required.

A more efficient method of applying machine learning to edge devices would be especially utile for visual tasks such as motion segmentation, object recognition, object classification, and pose estimation, which generally require the invocation of very high-cost networks such as Convolutional Neural Networks (CNNs). In search of the tools to effect such a method, we turn our attention to the animal brain, which is capable of perceiving motion from optical data with high speed and extreme energy efficiency. Hyperdimensional (HD) computing is a framework which attempts to replicate the animal brain's method of encoding information sourced from a variety of disparate sensors in a standardized format [1], [2]. HD computing is designed for performing efficient, low-power, complex calculations by first encoding concepts and values into uniformly-sized, high-dimension vectors. These HD vectors are often sparse, and have the useful properties of being *holographic*, insofar as they distribute information across many bits and *redundant*, insofar as they repeat encoded information, two qualities that make them very robust against noise. They are also *type-agnostic*, allowing data from a variety of different modalities to be encoded in the same format, and ultimately bound together.

Here we address the following questions: How can the binding and bundling methods proposed by existing HD algebras be properly adapted to create a new HD encoding scheme tailored specifically for capturing the spatiotemporal information that defines event-based data? What could an HD classifier predicated on this encoding scheme look like, and how may it be used to perform the difficult real-world task of pose estimation on multiple objects in a scene captured by a monocular event camera?

All authors are associated with the Perception and Robotics Group, University of Maryland, College Park. Emails: {neal, cmparam9, fer, yiannis} @umiacs.umd.edu

II. Event-Based Data and HD Computing

A. Event-Based Data

Whereas traditional cameras record frames at some fixed frame rate and integrates input from all pixels *synchronously*, event camera record the polarity of logarithmic intensity changes at each pixel *asynchronously*, transmitting data not as a complete image but as a series of information packets called *events* each corresponding to a change in brightness at a particular spatiotemporal coordinate in the full *event stream* produced by the camera. The trigger rule for an event to be propagated is

$$||\log(I_{t+\delta t,\mathbf{x}})-\log(I_{t,\mathbf{x}})|| \geq \tau \tag{1}$$

where $(I_{t,\mathbf{x}}$ is the brightness of a pixel at timestamp t and pixel location $\mathbf{x}$, δt is some small time increment, and τ is the the *trigger threshold* beyond which an intensity change is considered significant. Events can be formulated as tuples of the form $(\mathbf{x},t,p)$ where $p \in \{-1,1\}$ denotes the direction of brightness change of the pixel. The entire event stream produced by an event camera over some window is denoted by $(t,t+\delta t) = \{e_i\}_{i=1}^{N}$ [5]. In this paper, we explore an encoding scheme predicated on the bipolar VSA, i.e. a

VSA over a space of bipolar N-hypervectors $\mathbb{H} = \{v | v \in \{-1,1\}^D\}$; this VSA has the desirable property of being a "pure" VSA wherein bits are reduced to the simplest possible form, a single integer. In addition to this property, unlike many conventional VSAs this bipolar VSA has operations which preserve the sparsity [2] of the hypervectors involved, avoiding issues with vector degeneration and allowing for sparse preservation of information conducive to low-power computations on constrained hardware.

There are two fundamental operations associated with HD algebra, and we briefly define these operations in the general case before providing our particular implementations of them below:

B. Binding with the Discrete Fourier Transform

Hypervector *binding* is the HD analogue for multiplication, and is used to associate two hypervectors X,Y with one another as a key-value pair. For our purposes, this pair of vectors is made up of a *position vector* which encodes the "location" in some sense of a piece of information and a *description vector* which encodes the actual information found at this position. The process of encoding position and description in the context of spatiotemporal data is explained in detail in the following section.

Our bipolar VSA uses the following binding scheme predicated on the discrete Fourier transform to associate two equal-dimension hypervectors:

$$XY = \mathscr{F}^{-1}(\mathscr{F}(X) \circ \mathscr{F}(Y))$$

Where $\mathscr{F}$ denotes the Fourier transform, and $\circ$ denotes the computationally-efficient Hadamard product, which by the Fourier convolution theorem is equivalent to convolution in the Fourier domain. This binding method is specifically chosen out of various potential methods including the tensor product because of the significance of the Fourier transform in the computation of the brain—it has been observed that the collation of information in the neurons of the entorhinal cortex (responsible for, among other things, memory, perception, and navigation) can be described mathematically by a formulation similar to the inverse Discrete Fourier Transform [9].

This binding method preserves the bipolarity and sparseness of X and Y in their product, making these properties invariants of the VSA. We also present the complementary unbinding operation, accomplished by binding a vector that is itself the binding of two other vectors with one of its component vectors to retrieve an exact reconstruction of the other component vector:

$$Z = XY \implies X^{-1}Z = Y$$

Where X^{-1} denotes the inverse of X with respect to convolution, i.e.

$$X^{-1} = \mathscr{F}^{-1}(\mathscr{F}(X))$$

It is also possible to use the pseudoinverse X^{+} of X with respect to circular convolution to obtain an approximate reconstruction of Y from Z, which is useful in cases where optimization is needed rather than exact retrieval, such as in target propagation.

C. Bundling with the Stochastic Sign Function

Hypervector *bundling* is the HD analogue for addition, and is used to collect an indefinite number of hypervectors $X_1, X_2, X_3, \ldots$ into a single *HD memory* which encodes some number of related vectors, such as a population of class vectors in the case of *HD associative memory* or a population of samples with some discrete feature value in the case of *HD item memory*.

Our bipolar VSA uses the following bundling scheme, which preserves bipolarity and approximately preserves vector sparsity: an arbitrary number of vectors can be bundled to produce a vector composition

$$Y = sgn(\textstyle\sum_{i=1}^{D} X_i)$$

where $sgn(z)$ is the stochastic sign function, which returns 1 when $z > 0$, -1 when $z < 0$, and 1 or -1 with equal probability when $z = 0$. The stochastic nature of the bundling operation introduces random noise, but the aforementioned noise resistance of VSA vectors make them robust against this and empirical tests show component vectors can be queried with very high accuracy through unbinding performed on long vector bundles comprised of thousands of component vectors, when the hypervector dimension is high enough [2].

III. Encoding Spatiotemporal Data in HD

An arbitrary hyperdimensional computing framework may be formulated as the product of some *codebook*, Ψ, such that any given identifier, value, function, or other article of information x can be translated into high-dimensional space as a unique vector $v = \Psi(x)$. To draw an analogy to the realm of conventional CNNs, a codebook can be understood as the HD computing analogue to a convolutional kernel or filter—like a kernel, the codebook is permuted with input data to produce meaningful encoded patterns, and, thus, the choice of what codebook to use in a given HD computing paradigm is of great importance.

Typically, codebooks are created by sampling some distribution over the HD space $\mathbb{H}$ or some subspace thereof [3], [4]. The identity of the distribution sampled and the details of the encoding method determine a) the properties of encoded values that are preserved through encoding and b) the degree to which relations between encoded values are preserved after encoding. The ideal codebook for spatiotemporal data preserves the maximal amount of information with respect to spatial and temporal patterns in data.

We propose a novel HD encoding scheme intended specially for encoding dense DVS event cloud data while retaining spatiotemporal patterns. We process event cloud data by segmenting each cloud into a short event stream (generally < 0.2 seconds in length) and then passing a small (generally $< 4x4$ pixels over the spatial axes) kernel over the stream to calculate the average surface normal vector of the surface contained in that segment of the stream. These

TABLE I

COMMON HD ENCODING SCHEMES

Name	Mechanism	Dim
Uniform [3]	All values random	0
Thermometric [3], [4]	Higher values denser	1
Permutational [7]	Permuted positions	2
Moving Matrix Binding [8]	Additive matrix positions	2
Fractional Power Encoding [2]	Exponentiated positions	2

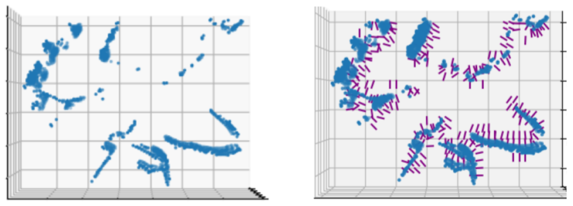

Fig. 1. Left: An event data slice containing the spatiotemporal surface generated by some object moving through space (EV-IMO [10]), in blue. Right: The local surface normal vectors computed along this spatiotemporal surface, in violet.

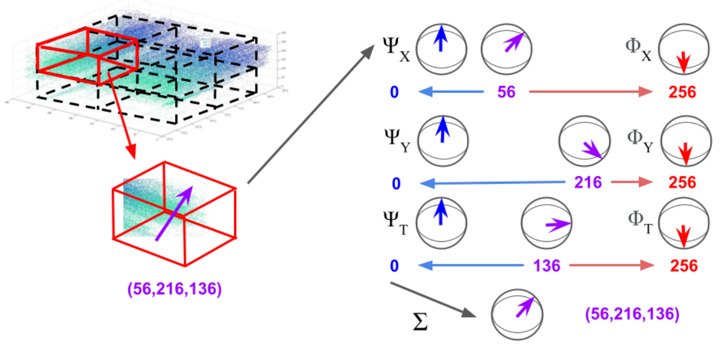

Fig. 2. Left: A region centered at $(56,216,136)$ is selected from a spatiotemporal surface and fitted using conventional means with a surface normal vector. Right: Limit interpolation is used to encode this vector component-wise into an HD vector.

average surface normals represent the approximate surface direction of event stream in that region, and by using a small stride (generally < 3 pixels) the total population of average surface normals accurately and succinctly describes the spatiotemporal contours and patterns inherent in this subset of the event cloud data. An example of these vectors is displayed in Figure 1.

The core encoding mechanism of our proposed encoding scheme is called *limit interpolation*. This process encodes a given value x along a dimension as an interpolation of limit vectors, i.e. a point on the spectrum of vectors produced by replacing gradually more and more bits of Ψ with bits of Φ until the point at which 100% of the bits have been replaced. Vectors closer to either end of this spectrum will have low Hamming distances from the limit vector at that end, and high Hamming distances from the opposite limit vector, which retains positional correlation between positions close to one another along the range of a dimension. Explicitly, coordinate x in a given dimension is normalized to a distance $q \in [0,1]$ by dividing it by the total length of the range, and the last $q\%$ of the bits of the vector encoding that value are taken from the upper limit vector Φ, the rest from the lower limit vector Ψ. The equation representing this interpolation can be formulated as $I(q) = q\%[\Psi] + (1-q\%)[\Phi]$.

Each component (i.e. (X,Y,T)) of an average surface normal is encoded according to the above limit vector interpolation method based on uniquely generated limit vectors that define the limits along that component's axis. These component vectors are bound to unique pseudoorthogonal label vectors corresponding to each axis and bundled into a final vectorized average surface normal, which we call the *description vector* for its small region of the event stream. The description vector for a cloud stream with axes X,Y,T is calculated as $v = X' * I(x) + Y' * I(y) + T' * I(t)$, where the average surface normal is $< x,y,t >$, $I(v)$ is the interpolation function, and X', Y', T' are label vectors for each axis.

A *position vector* is also created to denote the position of the kernel at the time a particular average surface normal was created. This position vector encodes the pixel coordinates of the kernel in precisely the same manner as above (the time coordinate does not need to be encoded because every description vector is formed based on the entirety of the time axis over the spatial region it describes, making our encoding scheme time-agnostic). This position vector is bound to the description vector to produce the information vector that describes the cloud stream at one small region, and the bundling of all these vectors is our vectorized encoding of the stream as a whole. A visualization of this process is presented in Figure 2.

IV. AN HD CLASSIFIER FOR POSE ESTIMATION

Finally, we sketch out how our proposed HD encoding scheme can be used to construct an HD classifier intended for working directly with event-based data in practice. We explore in this section a simple architecture comprised of an HD encoder and a single-layer HD classifier designed for the real-world task of pose estimation [11], which we term *PoseHD*. For the sake of simplicity, we formulate pose in terms of 3-dimensional spatial translation, but this model can be extended in a straightforward way to include rotational motion and other quantities, provided that they are incorporated into the codebook. Diagrams and plots are drawn from preliminary experiments using the EV-IMO dataset [10].

During training (Figure 3), the event stream is first segmented into individual objects before being discretized into separate dense event slices containing the spatiotemporal time surface associated with a single object over some interval. This time surface is fitted with a population of normal vectors which capture the spatiotemporal contours of the object's path, and these normal vectors are subsequently encoded using the aforementioned spatiotemporal encoding scheme to produce description vectors for each

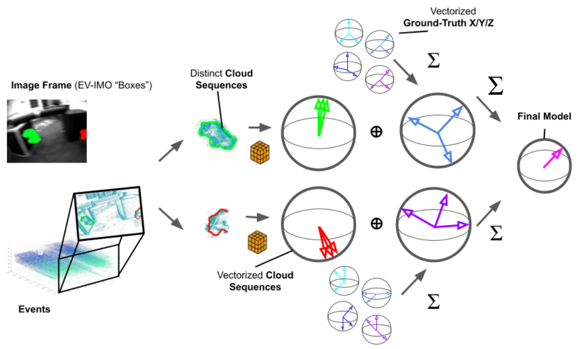

Fig. 3. Training pipeline for *PoseHD*.

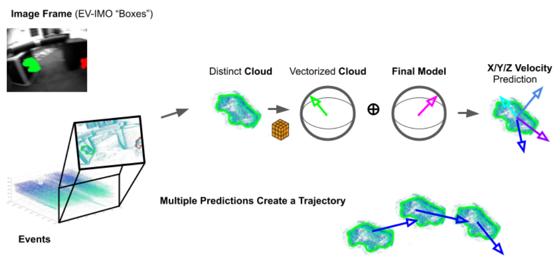

Fig. 4. Inference pipeline for *PoseHD*.

sector of the time surface. The description vectors are bound to their corresponding position vectors, and these position-description bindings are bundled to produce a hypervectorized representation of the event slice which captures the rich spatiotemporal contours of the object's path in a single vector. The translation space is discretized across each dimension, and the set of sample vectors which correspond to a certain discretized class vector are then bundled, producing the final associative memory.

During inference (Figure 4), the event slices selected from the test sequence are hypervectorized by the HD classifier and HD binding is performed between these sample hypervectors and the model's associative memory in order to query this memory. The output of this memory is a queried prediction hypervector which can itself be queried with the codebook description vectors associated with each pose dimension to determine the model's estimate of pose for the sample event cloud in a given dimension. Multiple queries generated in this way for adjacent slices can produce a pose trajectory.

V. Conclusion

We have proposed an HD encoding scheme, predicated on the favorable properties of the bipolar VSA and the Fourier transform, that is designed specifically for encoding spatiotemporal information. We illustrated the manner in which spatiotemporal surfaces may be described by fitted surface normals, and demonstrated how a large quantity of these normals can be encoded under our scheme and bundled into atomic hypervectorized representations of spatiotemporal surfaces. We closed with some discussion of how this encoding scheme may be incorporated into an HD classifier for the purpose of pose estimation of moving objects.

Further experiments are required to fully evaluate the efficacy of our encoding scheme across different event-based datasets and for disparate vision tasks. Further investigation is also required into alternate methods of encoding spatiotemporal surfaces—there are more thorough and accurate methods of fitting geometries to point clouds than local surface normal fitting, but it remains to be determined whether or not the tradeoff between computational cost and accuracy gain is worthwhile.

References

[1] Kanerva, P. (2009). Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. Cognitive computation, 1, 139-159.
[2] Frady, E. P., Kleyko, D., Kymn, C. J., Olshausen, B. A., Sommer, F. T. (2021). Computing on functions using randomized vector representations. arXiv preprint arXiv:2109.03429.
[3] Thomas, A., Dasgupta, S., Rosing, T. (2021). Theoretical Foundations of Hyperdimensional Computing. Journal of Artificial Intelligence Research, 72, 215-249.
[4] Kleyko, D., Kheffache, M., Frady, E. P., Wiklund, U., Osipov, E. (2020). Density encoding enables resource-efficient randomly connected neural networks. IEEE Transactions on Neural Networks and Learning Systems.
[5] Parameshwara, C. M., Li, S., Fermüller, C., Sanket, N. J., Evanusa, M. S., Aloimonos, Y. (2021). SpikeMS: Deep Spiking Neural Network for Motion Segmentation. arXiv preprint arXiv:2105.06562.
[6] Kleyko, D., Rachkovskij, D. A., Osipov, E., Rahimi, A. (2021). A Survey on Hyperdimensional Computing aka Vector Symbolic Architectures, Part I: Models and Data Transformations. arXiv preprint arXiv:2111.06077.
[7] Mitrokhin, A., Sutor, P., Summers-Stay, D., Fermüller, C., Aloimonos, Y. (2020). Symbolic representation and learning with hyperdimensional computing. Frontiers in Robotics and AI, 7, 63.
[8] Gallant, S. I., Culliton, P. (2016, August). Positional binding with distributed representations. In 2016 International Conference on Image, Vision and Computing (ICIVC) (pp. 108-113). IEEE.
[9] Orchard, J., Yang, H., Ji, X. (2013). Does the entorhinal cortex use the Fourier transform?. Frontiers in computational neuroscience, 7, 179.
[10] Mitrokhin, A., Ye, C., Fermüller, C., Aloimonos, Y., Delbruck, T. (2019, November). EV-IMO: Motion segmentation dataset and learning pipeline for event cameras. In 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 6105-6112). IEEE.
[11] Haralick, R. M., Joo, H., Lee, C. N., Zhuang, X., Vaidya, V. G., Kim, M. B. (1989). Pose estimation from corresponding point data. IEEE Transactions on Systems, Man, and Cybernetics, 19(6), 1426-1446.