A neural representation of continuous space using fractional binding

Brent Komer (bjkomer@uwaterloo.ca)
Terrence C. Stewart (tcstewart@uwaterloo.ca)
Aaron R. Voelker (arvoelke@uwaterloo.ca)
Chris Eliasmith (celiasmith@uwaterloo.ca)
Centre for Theoretical Neuroscience, University of Waterloo
Waterloo, ON, Canada, N2L 3G1

Abstract

We present a novel method for constructing neurally implemented spatial representations that we show to be useful for building models of spatial cognition. This method represents continuous (i.e., real-valued) spaces using neurons, and identifies a set of operations for manipulating these representations. Specifically, we use "fractional binding" to construct "spatial semantic pointers" (SSPs) that we use to generate and manipulate representations of spatial maps encoding the positions of objects. We show how these representations can be transformed to answer queries about the location and identities of objects, move the relative or global position of items, and answer queries about regions of space, among other things. We demonstrate that the neural implementation in spiking networks of SSPs have similar accuracy and capacity as the mathematical ideal.

Keywords: Semantic Pointer Architecture; spatial semantic pointer; spatial representation; fractional binding; continuous spaces; spiking neural networks

Introduction

There is evidence for a wide variety of types of mental representation. Some mental representations are well-described using *discrete* structures (e.g., graphs, trees, lists, and so on). Others are well-described using *continuous* structures (e.g., images, maps, surfaces, and so on). Here we propose a kind of mental representation of continuous structures that is amenable to neural implementation.

Recently, there have been several proposals for how neural networks can represent discrete structures. One family of approaches, called Vector Symbolic Architectures (VSAs), defines algebras over high-dimensional vector spaces, and uses those algebras to encode such structures. VSAs have been used to characterize a variety of cognitive behaviours, including analogical reasoning (Plate, 1994), language processing (Jones & Mewhort, 2007), and concept encoding (Crawford et al., 2015). Most VSAs are defined over continuous vector spaces, including Multiply Add Permute (MAP; Gayler, 2004), Holographic Reduced Representations (HRR; Plate, 1995), and Vector-derived Transformation Binding (VTB; Gosmann, 2018). When VSAs are used to model cognitive behaviours, they essentially define methods for characterizing continuous vectors as both slots and fillers and define a method of binding fillers to slots.

In this work, we will use the Semantic Pointer Architecture (SPA; Eliasmith, 2013), which proposes a means of neurally implementing VSAs for explaining cognitive behaviour in biologically plausible spiking networks. This architecture uses aspects of VSAs for cognitive representation, but the SPA also addresses visual processing, motor control, memory, decision making, and cognitive control in ways that do not use VSAs. However, all of these elements of the SPA use representations called semantic pointers (SPs), which result from compressing and decompressing information in cortex. As a result, we can think of VSA algebras as proposing a family of compression operators that are well-suited for certain cognitive tasks.

However, as with most uses of VSAs, in past work the SPA addresses cognitive tasks with a focus on representations of discrete structures (i.e., discrete slots in a represented structure). Here we propose a method for encoding cognitive structures over continuous spaces. We call this kind of representation "spatial semantic pointers" (SSPs). In this paper we propose and examine in some detail a specific kind of SSP implemented using a particular "fractional binding" operator to encode real-valued quantities – although a variety of other operators can be analogously defined.

In the remainder of the paper we provide a mathematical definition of SSPs, and show how SSPs can provide a natural means of generating and manipulating representations that are useful for spatial cognition. We identify desiderata for spatial representation that are useful for cognitive explanations. We then implement this representation both mathematically and neurally, and perform simulation experiments to demonstrate that it has a variety of useful properties, including: being able to query a memory for its spatial or non-spatial contents, representing multiple objects and locations simultaneously, spatially transforming memory contents without decoding them, and representing regions of space of various shapes and sizes. The choice of VSA and binding operator used in this work allows the representation and various transformations to be implemented efficiently by a spiking neural network.

A spatial representation

Our proposed representation generalizes the notion of vector binding to continuous spaces. By analogy to fractional powers defining the multiplication of reals, we define fractional bindings for vectors in a vector space. To explain, let us first consider binding a vector to itself a discrete number of times. That is, let $k \in \mathbb{N}$ be a natural number, $B \in \mathbb{R}^d$ be a fixed d-dimensional vector (i.e., semantic pointer), and $\circledast$ be a binding operator. We can repeatedly bind B with itself $k-1$

times[1] as follows:

$$B^k = \underbrace{B \circledast B \circledast \ldots \circledast B}_{B \text{ appears } k \text{ times}}. \quad (1)$$

This representation has been used in several cognitive models, for instance, to encode the position (k) in a list in serial working memory (Choo & Eliasmith, 2010). We propose to generalize this to continuous quantities (as opposed to discrete lists, for example) by permitting k to be real. Allowing a real k means that the resulting vector B^k encodes a continuous quantity. Most VSA operators can be interpreted in this manner (including MAP (Gayler, 2004), VTB (Gosmann, 2018), and HRR (Plate, 1995)), but not all (e.g., spatter codes (Kanerva, 1994)).

In the specific case of the SPA, we take the binding operator to be circular convolution (as proposed by Plate) and the fixed d-dimensional vectors to be semantic pointers chosen from the unit sphere. We then define our fractional binding operation by expressing equation 1 in the complex domain:

$$B^k = \mathcal{F}^{-1}\left\{\mathcal{F}\{B\}^k\right\}, \quad k \in \mathbb{R}, \quad (2)$$

where $\mathcal{F}\{\cdot\}$ is the Fourier transform, and $\mathcal{F}\{B\}^k$ is an element-wise exponentiation of a complex vector—analogous to exponentiation using fractional powers (e.g., $b^{2.5}$)—permitting k to be real.[2] In the present paper, we use unitary vectors for B due to the fact that their length does not change with multiple bindings, and their inverse is equal to their approximate inverse (see below).

This definition comes with many useful algebraic properties analogous to the relationship between multiplication and exponentiation (e.g., $b^{2.5}b^{1.5} = b^4$), in particular:

$$B^{k_1} \circledast B^{k_2} = B^{k_1+k_2}, \quad k_1, k_2 \in \mathbb{R}. \quad (3)$$

In essence, fractional binding is to circular convolution as exponentiation is to multiplication. We exploit equation 3 to perform semantically meaningful operations (e.g., shifting space) in our experiments.

Next, we extend this representation to multiple dimensions, which is the focus of our experiments below. In general, we can represent points in $\mathbb{R}^n$ by repeating equation 2, n times, using a different semantic pointer for each represented dimension (i.e., for each axis), and then binding all of the resulting vectors together. For $n = 2$, we think of the representation as encoding a continuous 2-D spatial representation (e.g., the location of objects on a map). In this case, the SSP that represents the point (x, y) is defined as the vector resulting from the function:

$$S(x, y) = X^x \circledast Y^y, \quad (4)$$

where X and Y are fixed semantic pointers, x and y are reals, and we are using fractional binding as defined by equation 2.

Similarly, the SSP that represents a continuous region (e.g., a solid rectangle), specified by some infinite set of 2-D points R, is defined as:

$$S(R) = \int_{(x,y)\in R} X^x \circledast Y^y \, dx\, dy. \quad (5)$$

There are efficient ways to compute equations 4 and 5 with spiking neurons using the Neural Engineering Framework (NEF; Eliasmith & Anderson, 2003). We use a publicly-available implementation in several of our results below.

To represent a single object occupying some location or region, we bind its semantic pointer representation, OBJ, with the SSP from equation 4 or 5, respectively:

$$M = OBJ \circledast S. \quad (6)$$

In general, to represent a set of m labelled objects together in the same memory, we can use superposition:

$$M = \sum_{i=1}^{m} OBJ_i \circledast S_i, \quad (7)$$

with a distinct semantic pointer OBJ_i tagging each object.

Given a representation like that in equation 7, we can query it in a number of ways. For example, to determine what object is at location (x, y) we can compute:

$$M \circledast (X^x \circledast Y^y)^{-1} = M \circledast X^{-x} \circledast Y^{-y}. \quad (8)$$

By the properties of binding and superposition, the resulting vector will have highest cosine similarity (i.e., dot product) with the object at (x, y).[3] Note that the inverse used in equation 8 is approximate, but choosing X and Y to be unitary vectors guarantees it is equal to the true inverse.

We can construct a heatmap of representations defined by equation 7, to visualize a decoding of the objects back into the original continuous space. For instance, for $m = 2$ (i.e., two represented objects), taking the dot product of $M \circledast OBJ_i^{-1}$ (M is from equation 7) with vectors representing positions spaced by Δx and Δy to tile the 2-D space, provides the visualization of Figure 1.

In summary, fractional binding provides a scheme for encoding a set of n-dimensional points into a d-dimensional SSP. This comes with an algebra for operating on these SSPs in meaningful ways (e.g., querying, shifting, and so on). When combined with the methods of the SPA, we can spatially manipulate collections of objects in a spiking neural implementation, as detailed in our experiments below.

Desiderata for spatial representation

To test if the proposed metric representation is useful, we consider its ability to be used in a variety of ways for representing, querying, and updating representations of objects in a spatial map. Here we describe the tests we perform, and in the next section we present the results of these tests.

[1] When $k = 0$ we get the identity vector corresponding to $\circledast$.

[2] For natural k, equations 1 and 2 are mathematically equivalent.

[3] This assumes d is sufficiently large, relative to m, as is typical for VSAs.

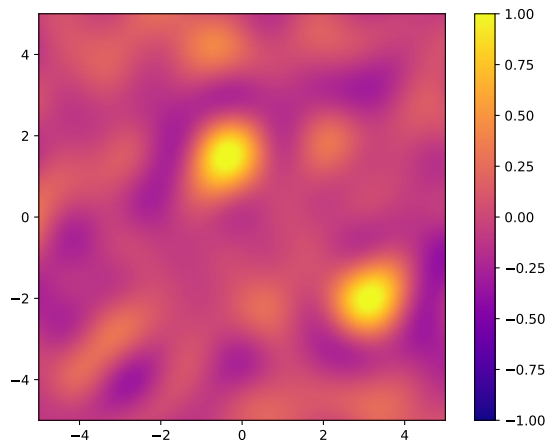


Figure 1: A heatmap visualizing the representation of two objects at different locations, as specified by equation 7. This graph is the sum of the decoding for each object.

The proposed desiderata for manipulating 2-D spatial representations are identified and briefly described in Table 1. In this table, M refers to the representation generated as described in the previous section.

Example queries

To illustrate the application of representing objects at continuous spatial locations using SSPs we demonstrate a variety of example queries in Figure 2. A set of animals (items) at random spatial locations are encoded into an SSP using equation 7 as shown in Figure 2a. This is accomplished by binding the SP representing each object with the SSP corresponding to its location, summing these values together, and then normalizing the result.

Various queries can be made with this representation. Figure 2c (top) shows the results of asking for the locations of different objects, decoded as a heatmap. If the object exists at more than one location, the resulting SSP will be highly similar to all of these locations (image on the left). If the object does not exist at any location, the resulting SSP will not be similar to any location on the heatmap (image on the right). Figure 2c (bottom) shows the reverse is possible too: given a location, find out which objects are at that location. If there are no objects at the queried location, the result will be noise and will not be similar to any object in the vocabulary (as shown in the far right).

Location queries can also be extended to regions of space, as shown in Figure 2b. If the region encompasses multiple objects, all objects should be returned, as depicted by the bar charts at the bottom. The region semantic pointers themselves are a single vector that is formed by integrating over the spatial semantic pointers within the region and normalizing the result, as described by equation 5. This process creates a vector that has a high dot product similarity with all vectors within a particular area while having a low dot product with vectors outside the region. Two example represented regions are illustrated in the heatmaps at the top of the figure. It is important to note that due to the normalization, the dot product with the region vector and a single point within the region will decrease as the area of the region increases. The consequence of this fact is that the optimal threshold for detection is a function of the area.

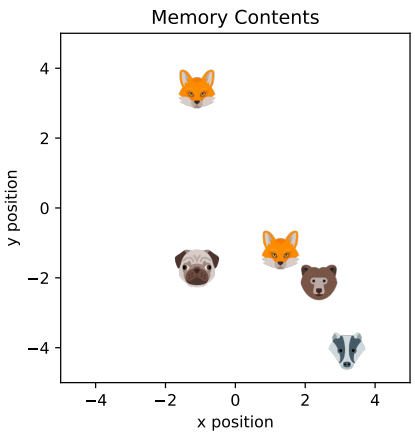
Desiderata	Description
Capacity	Determine how many objects can be encoded into M and a target object successfully decoded.
Query single object	Find the location of an object given the object and M.
Query missing object	Indicate if an object is not present when queried.
Query location	Determine what object is at a given location.
Query duplicate object	Determine the positions of multiple versions of the same object.
Neural implementation	Implement the operations in spiking neurons.
Region representation	Represent an entire region in the 2-D space.
Query Region	Determine which objects are in a spatial region.
Shift single object in group	Change the position of a single object without decoding M.
Shift whole group	Change the position of all objects in M similarly.
Readout (x,y) location from SSP	Map from the SSP representation to the 2-D space.

Table 1: Desiderata for metric representations of space.

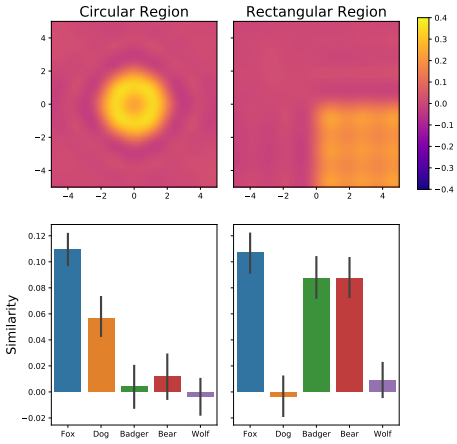
Experimental methods

To quantify how well this spatial representation performs in general for each of the desideratum a consistent measure must be used. In this paper we chose to use the accuracy of the output. When the output is a semantic pointer for an object, it is considered correct if its vector is more similar to the vector for the correct object than any other vector from a vocabulary of objects. Vocabularies are randomly generated semantic pointers of between 4 and 48 items, as described below. Similarity is determined by taking the dot product between vectors, with a larger value corresponding to a better match. When the output is a semantic pointer for a location, it is considered correct when the represented location is within 0.5 units of the true location. This threshold is chosen because it is approximately the radius of the region of similarity that a spatial semantic pointer has around itself.

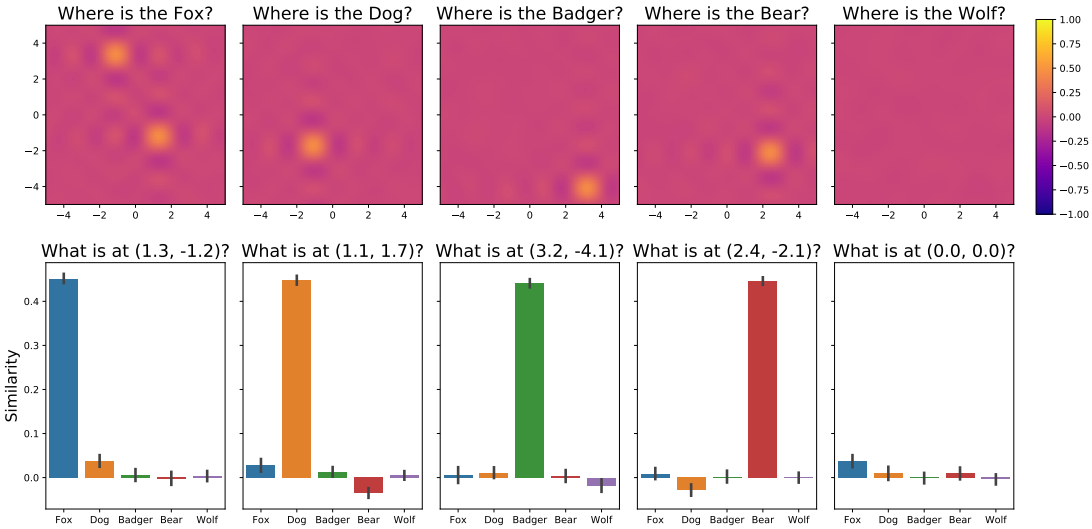
The capacity calculation requires identifying a threshold



(a) Memory contents



(b) Region queries



(c) Location and item queries

Figure 2: Example queries of items, locations, and regions. a) An example memory encoded into an SSP. b) Region queries applied to the memory in (a). c) Object (top) and location (bottom) queries applied to the memory in (a).

above which an item is identified as present in the representation. For this purpose we pick a threshold that is 3 sigma above the mean, ensuring a 99.7% chance of accepting only correct items.

For our capacity tests, we experiment with a dimensionality of 128, 256, and 512, and observe the overall effect on performance. In all other tests we fix the number of dimensions to 512, which we have found achieves good performance across a wide variety of tasks in both spiking and non-spiking regimes.

For each desideratum, accuracy reported is the mean across 6,000 trials with memory sizes varying uniformly between 2 and 24 items. For each trial the vocabulary of objects is chosen to be twice the number of objects encoded into memory (i.e., 4 to 48).

Each object is assigned a random unitary vector. All semantic pointers used in each task are normalized after every operation. All 2-D coordinates used in the experiments are chosen uniformly at random within the domain of -5 to 5 for both x and y. The size of the domain in relation to the dimensionality of the semantic pointers determines the ideal level of performance (not shown).

Query single object Equation 9 is used to produce an SSP representing the location of the desired object. Accuracy is computed by decoding this high-dimensional vector, S, into the 2-D coordinate it represents and comparing to the true location.

$$S = M \circledast OBJ^{-1}. \tag{9}$$

Query missing object Given a memory containing objects, query an object that does not exist (using equation 9). The correct behaviour is a result that is highly dissimilar to all

locations within the domain of interest. This is determined by the dot product of S and every SSP being less than 0.1.

Query duplicate object Given a memory containing many objects with some duplicates, query an object that appears twice. The correct behaviour is to return a spatial semantic pointer that represents both locations of this object.

Query location Use equation 8 with the location for one of the objects in memory. The correct behaviour is to return a semantic pointer for the object at that location.

Query region On each trial a circular region is created with a radius between 1 and 3 units and centered at a random location. An SSP is constructed for this region using equation 5. The inverse of this SSP is convolved with the memory to obtain a semantic pointer representing all objects in the region. Accuracy is computed by adding the number of objects correctly detected in the region to the number of objects correctly not detected from outside the region and then dividing by the total number of objects in the memory.

Shift single object in group Moving a single object within a group can be accomplished by adding the object of interest convolved with a vector that is the difference between the start and end positions, as shown in equation 10. Accuracy is reported for all objects as well as just the object that was moved.

$$\Delta M = OBJ \circledast \Delta S. \tag{10}$$

Shift whole group The memory is convolved with an SSP that corresponds to a random displacement, which leverages the property of equation 3. An object query is then performed for each object in the memory and the result is considered correct if it moved by the displacement amount. A heatmap visualizing the result of the two shifting operations is shown in Figure 3 for a group of three identical objects.

Readout (x,y) location from SSP For the non-neural case location is extracted from the maximum point in the heatmap. In spiking neurons a heteroassociative memory is optimized to map from a 512-dimensional SSP to a 2-D location.

Construct SSP from (x,y) location This can be computed directly from equation 4. For the experiment using spiking neurons each axis is first computed separately and then convolved together.

All experiments were repeated using networks of leaky integrate-and-fire (LIF) neurons and the NEF to implement the necessary transformations. In all trials 50 neurons were used per dimension to represent the memory and to compute circular convolutions.

Results

The results of the experiments for each of the desideratum are shown in Table 2.[4] As can be seen from the table, the SSP representation is able to address the desiderata quite well, both in purely mathematical and neural implementations. The worst performance is evident in the shifting of a single object in a group. Specifically, the accuracy of the representation for the objects that were not shifted decreases, while the accuracy for the shifted object increases. This is due to normalization effects making the moved object be re-encoded with a larger relative magnitude than the rest of the items. Using a scaling factor proportional to the number of items in the memory mitigates this effect (improves accuracy from 75.7% to 97.8%), but in general the number of items within a memory is not always known without first retrieving items from memory, and equation 10 is agnostic to the other contents of the memory.

To better characterize the capacity of a single memory using this representation we performed queries on memories with progressively larger numbers of items encoded (see Figure 4). The shape of the curve is very similar for both location and object queries since the decrease in the dot product is mostly a result of the normalization of the memory to a unit vector. The standard deviation for the dot product of two

[4]All source code required to reproduce these experiments and generate the figures is available at https://github.com/ctn-waterloo/cogsci2019-ssp.

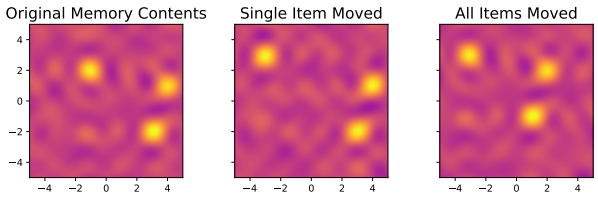


Figure 3: Shifting objects in memory. Left: The original memory. Middle: Shifting the top left object. Right: Shifting all three objects.

Desiderata	Accuracy	
	Non-Neural	Neural
Query single object	99.1%	95.7%
Query missing object	99.4%	96.7%
Query location	97.3%	94.7%
Query duplicate object	97.4%	95.3%
Query region	90.4%	73.5%
Shift single object in group (all objects)	75.7%	67.3%
Shift single object in group (moved object)	100.0%	100.0%
Shift whole group	97.8%	96.7%
Readout (x,y) location from SSP	100.0%	94.1%
Construct SSP from (x,y) location	100.0%	99.0%

Table 2: Experiments for the desiderata for metric representations of space. Accuracy is calculated using SSP representations containing 2 to 24 items. When the output is a location, it is considered correct when the result is within 0.5 units of the true location.

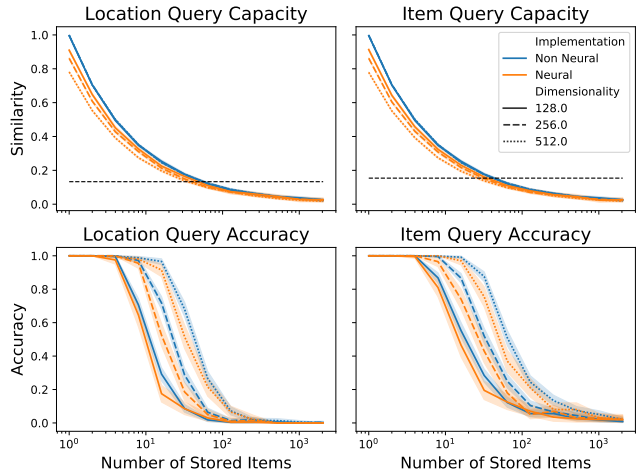


Figure 4: Memory capacity and accuracy as a function of the number of items in the SSP for ideal and neural implementations while varying the dimensionality. Top panels show the item and location capacity. Bottom panels show the item and location accuracy.

random vectors in a unit hypersphere is $\sqrt{1/D}$ where D is the dimensionality of the space. The mean is zero, so for 512 dimensions this results in a 3-sigma threshold similarity of 0.133. SSPs that represent coordinates within a finite domain will span a smaller subspace of the hypersphere, so their threshold will be a little higher. Specifically, we estimated the threshold by generating 10,000 random SSPs from a 10×10 2-D domain and computing the dot product between every pair. The mean is approximately zero, and three standard deviations is 0.154. Consequently 99.7% of queries will be above this value for items actually in the memory. The accuracy plots show the importance of dimensionality on accuracy of decoding memories.

Discussion

We have proposed a novel neural representation, SSPs, for encoding structured continuous spaces using fractional binding. We have demonstrated that these representations satisfy desiderata for representations that are useful for spatial cognition. By implementing these methods at the level of spiking neurons, this work enables future exploration of tradeoffs between neural constraints and performance for tasks of increasing complexity. In addition, a spiking neural implementation serves as a prerequisite for constructing dynamical models of spatial cognition that operate sparsely over time and in an event-driven manner.

SSPs have many potential applications for modelling cognitive phenomena that involve spatial reasoning over time, such as path planning and navigation. Objects that a cognitive agent encounters while traversing a space can be stored in memory in such a way that their relative distances and directions from each other are preserved.

The extension of Vector Symbolic Architectures to continuous representation presented in this work is not limited to representing physical space. Any continuous dimension over which concepts may vary (e.g., mass, colour, value, and so on) can utilize this representation.

While we have explored some of the capacity and accuracy limitations of this representation, it is important to note that the effective capacity can likely be increased by hierarchically chunking items into groups when encoding them into the memory by using a similar technique as the method demonstrated in Crawford et al. (2015).

Areas of future work include exploring the theoretical foundations of this method to improve our understanding of its strengths and limitations. As well, there remain many questions regarding how well a cognitive model using these representations can scale and how well the behaviour and neural recordings from such a model match that of animals.

Acknowledgments

We would like to thank Jan Gosmann for his work on the mathematical foundations of fractional binding for semantic pointers, and personal discussions. This work was supported by CFI and OIT infrastructure funding, the Canada Research Chairs program, NSERC Discovery grant 261453, ONR grant N000141310419, AFOSR grant FA8655-13-1-3084, OGS, and NSERC CGS-D.

References

Choo, X., & Eliasmith, C. (2010, 08/2010). *A spiking neuron model of serial-order recall.* Portland, OR: Cognitive Science Society.

Crawford, E., Gingerich, M., & Eliasmith, C. (2015). Biologically plausible, human-scale knowledge representation. *Cognitive Science*. doi: 10.1111/cogs.12261

Eliasmith, C. (2013). *How to build a brain: A neural architecture for biological cognition*. Oxford University Press.

Eliasmith, C., & Anderson, C. H. (2003). *Neural engineering: Computation, representation, and dynamics in neurobiological systems*. MIT press.

Gayler, R. W. (2004). Vector symbolic architectures answer Jackendoff's challenges for cognitive neuroscience. *arXiv preprint cs/0412059*.

Gosmann, J. (2018). An integrated model of context, short-term, and long-term memory.

Jones, M. N., & Mewhort, D. J. (2007). Representing word meaning and order information in a composite holographic lexicon. *Psychological review*, *114*(1), 1.

Kanerva, P. (1994). The spatter code for encoding concepts at many levels. In *Icann94* (pp. 226–229). Springer.

Plate, T. A. (1994). Estimating analogical similarity by dot-products of holographic reduced representations. In *Advances in neural information processing systems* (pp. 1109–1116).

Plate, T. A. (1995). Holographic reduced representations. *IEEE Transactions on Neural networks*, *6*(3), 623–641.