LETTER **Communicated by Simon Levy**

Vector-Derived Transformation Binding: An Improved Binding Operation for Deep Symbol-Like Processing in Neural Networks

Jan Gosmann
jgosmann@uwaterloo.ca
Chris Eliasmith
celiasmith@uwaterloo.ca
Centre for Theoretical Neuroscience, University of Waterloo, Waterloo, ON N2L 3G1 Canada

We present a new binding operation, vector-derived transformation binding (VTB), for use in vector symbolic architectures (VSA). The performance of VTB is compared to circular convolution, used in holographic reduced representations (HRRs), in terms of list and stack encoding capacity. A special focus is given to the possibility of a neural implementation by the means of the Neural Engineering Framework (NEF). While the scaling of required neural resources is slightly worse for VTB, it is found to be on par with circular convolution for list encoding and better for encoding of stacks. Furthermore, VTB influences the vector length less, which also benefits a neural implementation. Consequently, we argue that VTB is an improvement over HRRs for neurally implemented VSAs.

1 Introduction

Many cognitive tasks require the brain to perform symbolic processing. How such symbolic processing could be implemented in the brain or an artificial neural network is, however, not obvious. Vector symbolic architectures (VSAs) have been proposed as a means to solve this problem while using high-dimensional vectors to represent symbols (Kanerva, 2009). It has been suggested that such systems can directly address Jackendoff's linguistic challenges, which he posed to cognitive neuroscience (Gayler, 2004).

VSAs associate (high-dimensional) vectors with specific symbols and use certain mathematical operations to manipulate these vectors in order to implement symbol-like processing. Three operations—an addition-like superposition or set operator, a multiplication-like binding operator, and a permutation-like hiding operator—are essential (Gayler, 2004). The binding operator may also act like a hiding operator to protect vectors from other operations, in which case a separate hiding operator is not needed. The exact choice of operators differs across various VSAs.

Neural Computation **31, 849–869** (2019)
doi:10.1162/neco_a_01179

For our purposes, a defining feature of any VSA is that the dimensionality of the vectors stays constant across all operations. This is in contrast to tensor products (Smolensky, 1990) that create an outer product for each binding, leading to quadratically increasing dimensionality. This poses a challenge to the biological plausibility of tensor products due to the excessive demands such dimensionality increase puts on neural resources (Eliasmith, 2013). While VSAs avoid this scaling problem, the more constrained dimensionality forces the binding operation to do a form of lossy compression. Accordingly, different binding operations may lose more or less information due to this compression. Ideally the amount of retained information should be maximized.

It has been argued that VSAs could be a step toward identifying mathematical operations and representational systems that mimic cognitive phenomena (Kanerva, 2009). In particular, VSAs have been used successfully to model compositionality and semantics (Mitchell & Lapata, 2010; Recchia, Sahlgren, Kanerva, & Jones, 2015), as well as a basis for spiking neural network models of various cognitive tasks. Such tasks include the n-back task (Gosmann & Eliasmith, 2015), the Tower of Hanoi task (Stewart & Eliasmith, 2011), human-scale knowledge representation (Crawford, Gingerich, & Eliasmith, 2016), and Spaun, a large-scale functional brain model capable of performing eight different tasks and capturing a wide variety of physiological and anatomical features of the mammalian brain (Eliasmith et al., 2012). Consequently, in this letter, we are interested in binding operations that perform best in the context of such models.

Compared to classic neural networks trained with backpropagation as used in the deep learning approach, VSAs have a number of advantages. They allow encoding of a structure with an explicit encoding scheme, whereas this encoding scheme will be implicit in a classic neural network. In addition, classic training methods can be time intensive and require many training examples. Furthermore, while VSA operations can be performed with neural networks, it is not a requirement. As a result, the calculations can also be evaluated directly in algebra, and to some degree, it is possible to perform mathematical analyses, as the operations are made explicit. However, emphasize that VSAs and more classical neural network approaches do not need to be opposed. In fact, they can be integrated as, for example, done in Spaun (Eliasmith et al., 2012) where the visual input is parsed with a system trained with deep learning principles, but the output is interpreted and used as a vector in a VSA.

We start by introducing general definitions and desired properties of operators in a VSA, followed by the definition of the specific binding operations of circular convolution and vector-derived transformation binding (VTB). In section 3, we discuss how these binding methods can be used to build up structured representation with a superposition of pairwise bindings or alternatively tagging. We then move on to compare the

two presented binding operations, looking at their binding properties, list encoding capacity, stack encoding capacity, and neural scaling. The results are discussed in section 5.

2 Vector Symbolic Architectures

We start by introducing the abstract operators required in a VSA. We begin by specifying a measure of similarity for the d-dimensional vectors being employed. As with most other VSAs, we use the normalized dot product (cosine similarity):

$$\begin{aligned} s: \ & \mathbb{R}^d \times \mathbb{R}^d \longrightarrow \mathbb{R} \\ & \boldsymbol{x}, \boldsymbol{y} \longmapsto \frac{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}{\|\boldsymbol{x}\| \cdot \|\boldsymbol{y}\|}. \end{aligned} \tag{2.1}$$

Turning to the three required VSA operators—a set, binding, and hiding operator—we can begin with the superposition or set operator,

$$\mathcal{S} = \mathbb{R}^d \times \mathbb{R}^d \longrightarrow \mathbb{R}^d, \tag{2.2}$$

which produces a vector similar to both operands—that is, $s(\mathcal{S}(\boldsymbol{x}, \boldsymbol{y}), \boldsymbol{x}) \approx s(\mathcal{S}(\boldsymbol{x}, \boldsymbol{y}), \boldsymbol{y}) \gtrapprox \sqrt{1/2}$ for $s(\boldsymbol{x}, \boldsymbol{y})$. This will usually be, and is for the remainder of this letter, simple elementwise addition: $S(\boldsymbol{x}, \boldsymbol{y}) := \boldsymbol{x} + \boldsymbol{y}$.

We will co-specify the binding and hiding operators, although these can be specified independently (Gayler, 1998). Co-specifying these reduces the operator count, making the algebra simpler. In addition, most neural implementations have used circular convolution, a binding operator that includes a permutation for hiding. We refer to such operators as binding operators.

In general, the binding operator can be specified as

$$\mathcal{B}: \mathbb{R}^d \times \mathbb{R}^d \longrightarrow \mathbb{R}^d \tag{2.3}$$

with an approximate inverse,[1]

$$\mathcal{B}^+: \mathbb{R}^d \times \mathbb{R}^d \longrightarrow \mathbb{R}^d, \tag{2.4}$$

which can be used as an unbinding operation as required. The binding and unbinding operations are required to be distributive,

[1] We use the superscript "plus" in analogy to a common notation of the matrix pseudo-inverse to emphasize the approximate nature in contrast to the exact inverse that also exists for some binding operations.

$$\mathcal{B}(\boldsymbol{x}+\boldsymbol{y},\boldsymbol{z}) = \mathcal{B}(\boldsymbol{x},\boldsymbol{z}) + \mathcal{B}(\boldsymbol{y},\boldsymbol{z}), \tag{2.5}$$

$$\mathcal{B}^{+}(\boldsymbol{x}+\boldsymbol{y},\boldsymbol{z}) = \mathcal{B}^{+}(\boldsymbol{x},\boldsymbol{z}) + \mathcal{B}^{+}(\boldsymbol{y},\boldsymbol{z}), \tag{2.6}$$

to allow the composition of structured representations (see section 3).

Furthermore, to be useful, two additional properties should be fulfilled by the binding operator. First, the unbinding operator should yield $\mathcal{B}^{+}(\mathcal{B}(\boldsymbol{x},\boldsymbol{y}),\boldsymbol{y}) \approx \boldsymbol{x}$ for almost all $\boldsymbol{x}$ and $\boldsymbol{y}$; that is, the recovery of a bound vector should be possible. Second, the bound result of the binding operator should be, in contrast to the superposition operator, dissimilar to both of the operands: $\boldsymbol{x} \not\approx \mathcal{B}(\boldsymbol{x},\boldsymbol{y}) \not\approx \boldsymbol{y}$ (this results from the operator "hiding" the results of the binding). Finally, in a neural system, the operators have to be robust against noise; that is, the amplification of the relative error through binding and unbinding should be limited by a constant C so that

$$\left\langle \frac{\|\mathcal{B}(\boldsymbol{x},\boldsymbol{y}+s\boldsymbol{\xi}) - \mathcal{B}(\boldsymbol{x},\boldsymbol{y})\|}{\|\mathcal{B}(\boldsymbol{x},\boldsymbol{y})\|} \right\rangle_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{\xi}} = \left\langle \frac{\|\mathcal{B}(\boldsymbol{x},s\boldsymbol{\xi})\|}{\|\mathcal{B}(\boldsymbol{x},\boldsymbol{y})\|} \right\rangle_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{\xi}} \leq Cs, \tag{2.7}$$

where $\boldsymbol{x},\boldsymbol{y},\boldsymbol{\xi} \in \mathbb{R}^d$ with $\|\boldsymbol{x}\| = \|\boldsymbol{y}\| = \|\boldsymbol{\xi}\| = 1$ and $s \geq 0$ is the scale of the noise. The same relation should hold when replacing $\mathcal{B}$ with $\mathcal{B}^{+}$ in equation 2.7.

Given a specific binding operation, certain special elements in $\mathbb{R}^d$ can be identified that are defined in the following manner:

Definition 1 *(identity vector). A vector $\boldsymbol{i}_{\mathcal{B}}$ with the property $\mathcal{B}(\boldsymbol{x},\boldsymbol{i}_{\mathcal{B}}) = \boldsymbol{x}$ is called identity vector under $\mathcal{B}$.*

Definition 2 *(absorbing element). A vector $\boldsymbol{n}_{\mathcal{B}}$ with the property $\mathcal{B}(\boldsymbol{x},\boldsymbol{n}_{\mathcal{B}}) = c \cdot \boldsymbol{n}_{\mathcal{B}}$ where $c \in \mathbb{R}$ is called an absorbing element under $\mathcal{B}$.*

Such an absorbing element effectively destroys the information in the vector $\boldsymbol{x}$. For that reason, absorbing elements should be avoided when constructing representations with binding. Note that this definition slightly differs from the usual definition of absorbing elements by allowing for a scaling factor.

Definition 3 *(unitary vector). A vector $\boldsymbol{u}$ with the property $\langle \mathcal{B}(\boldsymbol{x},\boldsymbol{u}), \mathcal{B}(\boldsymbol{y},\boldsymbol{u})\rangle = \langle \boldsymbol{x},\boldsymbol{y}\rangle$ is called unitary.*

In other words, a unitary vector preserves the dot product under binding. This is in analogy to unitary transformation matrices that also preserve the dot product. It also implies that binding with a unitary vector preserves the norm of the bound vector.

2.1 Circular Convolution. Plate (2003) proposed circular convolution as a binding operation with his holographic reduced representations (HRRs):

Definition 4 *(circular convolution binding). The circular convolution binding operator is given by*

$$\mathcal{B}_{\circledast}(\boldsymbol{x}, \boldsymbol{y}) := \boldsymbol{x} \circledast \boldsymbol{y} \text{ with } (\boldsymbol{x} \circledast \boldsymbol{y})_i = \sum_{j=1}^{d} x_j y_{((i-j) \bmod d)+1} \tag{2.8}$$

and has the approximate inverse (Plate 2003),

$$\mathcal{B}^{+}_{\circledast}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x} \circledast \boldsymbol{y}^{+} \text{ with } \boldsymbol{y}^{+} := (y_1, y_d, y_{d-1}, \ldots, y_2)^{\top}. \tag{2.9}$$

As an example, for $d = 4$ with the vectors $\boldsymbol{x} = (-0.8, 0.08, 0.04, -0.6)^{\top}$ and $\boldsymbol{y} = (0.97, 0.16, 0.03, -0.16)^{\top}$, $\mathcal{B}_{\circledast}(\boldsymbol{x}, \boldsymbol{y})$ works out to:

$$\begin{aligned}\mathcal{B}_{\circledast}(\boldsymbol{x}, \boldsymbol{y}) &= \begin{bmatrix} -0.8 \cdot 0.97 - 0.08 \cdot 0.16 + 0.04 \cdot 0.03 - 0.6 \cdot 0.16 \\ -0.8 \cdot 0.16 + 0.08 \cdot 0.03 - 0.04 \cdot 0.16 - 0.6 \cdot 0.97 \\ -0.8 \cdot 0.03 - 0.08 \cdot 0.16 + 0.04 \cdot 0.97 - 0.6 \cdot 0.16 \\ 0.8 \cdot 0.16 + 0.08 \cdot 0.97 + 0.04 + 0.16 - 0.6 \cdot 0.97 \end{bmatrix} \\ &= \begin{bmatrix} -0.8836 \\ -0.0748 \\ 0.1236 \\ -0.4452 \end{bmatrix}.\end{aligned}$$

A useful property of circular convolution is that it becomes an elementwise multiplication in the Fourier space. Thus, it can be written as

$$\boldsymbol{x} \circledast \boldsymbol{y} = \boldsymbol{F}^{-1}[(\boldsymbol{F}\boldsymbol{x}) \odot (\boldsymbol{F}\boldsymbol{y})] \tag{2.10}$$

given the discrete Fourier transform matrix $\boldsymbol{F}$. In essence, this mapping to the Fourier space acts as a hiding operator while the elementwise multiplication performs the binding. Given this formulation, it is easy to see that the basic properties of

$$\text{Distributivity:} \quad (\boldsymbol{x}_1 + \boldsymbol{x}_2) \circledast \boldsymbol{y} = \boldsymbol{x}_1 \circledast \boldsymbol{y} + \boldsymbol{x}_2 \circledast \boldsymbol{y}, \tag{2.11}$$

$$\text{Associativity:} \quad (\boldsymbol{x} \circledast \boldsymbol{y}) \circledast \boldsymbol{z} = \boldsymbol{x} \circledast (\boldsymbol{y} \circledast \boldsymbol{z}), \tag{2.12}$$

$$\text{Commutativity:} \quad \boldsymbol{x} \circledast \boldsymbol{y} = \boldsymbol{y} \circledast \boldsymbol{x}, \tag{2.13}$$

$$\text{Linearity of arguments:} \quad (a\boldsymbol{x}) \circledast (b\boldsymbol{y}) = ab(\boldsymbol{x} \circledast \boldsymbol{y}), \quad a, b \in \mathbb{R} \tag{2.14}$$

hold for circular convolution. From the linearity of the arguments, it follows that the amplification of relative error due to noise is limited with the constant $C = 1$.

The Fourier space formulation also makes the derivation of the special elements straightforward. The identity vector must not change any of the complex Fourier coefficients, and thus, all of its coefficients must be $1 + 0\mathrm{i}$, which yields

$$\boldsymbol{i}_{\circledast} = (1, 0, 0, \ldots, 0)^{\top}. \tag{2.15}$$

All vectors with Fourier coefficients of 0 except for the DC offset will be absorbing elements corresponding to the vectors

$$\boldsymbol{n}_{\circledast} = (z, \ldots, z)^{\top}, \quad z \in \mathbb{R}, \tag{2.16}$$

including the vector of all zeros $(0, \ldots, 0)^{\top}$. Finally, all vectors with Fourier coefficients $c_i \in \mathbb{C}$ that are of unit length $|c_i| = 1$ are unitary vectors, as the frequency amplitudes are not scaled.

2.2 Vector-Derived Transformation Binding. We propose a new binding operator that we call the vector-derived transformation binding (VTB):

Definition 5 *(vector-derived transformation binding, VTB). Given a dimensionality $d' = d^{1/2} \in \mathbb{N}_{>0}$, the vector-derived transformation binding operator $\mathcal{B}_{\mathrm{V}} : \mathbb{R}^d \times \mathbb{R}^d \longrightarrow \mathbb{R}^d$ is defined as*

$$\mathcal{B}_{\mathrm{V}}(\boldsymbol{x}, \boldsymbol{y}) := \boldsymbol{V}_{\boldsymbol{y}}\boldsymbol{x} = \begin{bmatrix} \boldsymbol{V}'_{\boldsymbol{y}} & 0 & 0 \\ 0 & \boldsymbol{V}'_{\boldsymbol{y}} & 0 \\ 0 & 0 & \ddots \end{bmatrix} \boldsymbol{x} \tag{2.17}$$

with

$$\boldsymbol{V}'_{\boldsymbol{y}} = d^{\frac{1}{4}} \begin{bmatrix} y_1 & y_2 & \cdots & y_{d'} \\ y_{d'+1} & y_{d'+2} & \cdots & y_{2d'} \\ \vdots & \vdots & \ddots & \vdots \\ y_{d-d'+1} & y_{d-d'+2} & \cdots & y_d \end{bmatrix}, \tag{2.18}$$

where y_i is the i-th vector component of $\boldsymbol{y}$. The approximate inverse is given by

$$\mathcal{B}^{+}_{\mathrm{V}}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{V}_{\boldsymbol{y}}{}^{\top}\boldsymbol{x} = \begin{bmatrix} V'^{\top}_{\boldsymbol{y}} & 0 & 0 \\ 0 & V'^{\top}_{\boldsymbol{y}} & 0 \\ 0 & 0 & \ddots \end{bmatrix} \boldsymbol{x}. \tag{2.19}$$

As an example, for $d = 4$ with the vectors $\boldsymbol{x} = (-0.8, 0.08, 0.04, -0.6)^\top$ and $\boldsymbol{y} = (0.97, 0.16, 0.03, -0.16)^\top$, $\mathcal{B}_{\mathrm{V}}(\boldsymbol{x}, \boldsymbol{y})$ works out to

$$\begin{aligned}\mathcal{B}_{\mathrm{V}}(\boldsymbol{x}, \boldsymbol{y}) &= \sqrt{2}\begin{bmatrix} 0.97 & 0.16 & 0 & 0 \\ 0.03 & -0.16 & 0 & 0 \\ 0 & 0 & 0.97 & 0.16 \\ 0 & 0 & 0.03 & -0.16 \end{bmatrix}\begin{bmatrix} -0.8 \\ 0.08 \\ 0.04 \\ -0.6 \end{bmatrix} \\ &= \sqrt{2}\begin{bmatrix} -0.97 \cdot 0.8 + 0.16 \cdot 0.08 \\ -0.03 \cdot 0.8 - 0.16 \cdot 0.08 \\ 0.97 \cdot 0.04 - 0.16 \cdot 0.6 \\ 0.03 \cdot 0.04 + 0.16 \cdot 0.6 \end{bmatrix} = \sqrt{2}\begin{bmatrix} -0.7632 \\ -0.0368 \\ -0.0572 \\ 0.0972 \end{bmatrix} \approx \begin{bmatrix} -1.08 \\ -0.05 \\ -0.08 \\ 0.14 \end{bmatrix}.\end{aligned}$$

This binding operator relies on the fact that the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ are usually picked randomly from a uniform distribution of unit-length vectors. That implies that the individual vector components are identically distributed and $\boldsymbol{V}_y$ is approximately orthogonal: ${\boldsymbol{V}_y}^\top \boldsymbol{V}_y \approx \boldsymbol{I}$. This makes the binding a random transformation of $\boldsymbol{x}$ based on $\boldsymbol{y}$, thus ensuring that the result will be dissimilar to both inputs (for most vectors as long as $\boldsymbol{x}$ does not happen to be an eigenvector of $\boldsymbol{V}_y$). The linearity of the matrix multiplication ensures that the amplification of the relative error is bounded with $C = 1$. Also, the distributivity requirement is fulfilled, as one can easily verify.

Corollary 1 *(VTB distributivity). VTB is distributive:*

$$\begin{aligned}&\mathcal{B}_{\mathrm{V}}(\boldsymbol{x}_1 + \boldsymbol{x}_2, \boldsymbol{y}) = \mathcal{B}_{\mathrm{V}}(\boldsymbol{x}_1, \boldsymbol{y}) + \mathcal{B}_{\mathrm{V}}(\boldsymbol{x}_2, \boldsymbol{y}) \text{ and} \\ &\mathcal{B}_{\mathrm{V}}(\boldsymbol{x}, \boldsymbol{y}_1 + \boldsymbol{y}_2) = \mathcal{B}_{\mathrm{V}}(\boldsymbol{x}, \boldsymbol{y}_1) + \mathcal{B}_{\mathrm{V}}(\boldsymbol{x}, \boldsymbol{y}_2).\end{aligned} \tag{2.20}$$

Proof. By applying the definitions for both directions of the distributivity,

- $\mathcal{B}_{\mathrm{V}}(\boldsymbol{x}_1 + \boldsymbol{x}_2, \boldsymbol{y}) = \boldsymbol{V}_y(\boldsymbol{x}_1 + \boldsymbol{x}_2) = \boldsymbol{V}_y\boldsymbol{x}_1 + \boldsymbol{V}_y\boldsymbol{x}_2 = \mathcal{B}_{\mathrm{V}}(\boldsymbol{x}_1, \boldsymbol{y}) + \mathcal{B}_{\mathrm{V}}(\boldsymbol{x}_2, \boldsymbol{y})$
- $\mathcal{B}_{\mathrm{V}}(\boldsymbol{x}, \boldsymbol{y}_1 + \boldsymbol{y}_1) = \boldsymbol{V}_{\boldsymbol{y}_1+\boldsymbol{y}_2}\boldsymbol{x} = (\boldsymbol{V}_{\boldsymbol{y}_1} + \boldsymbol{V}_{\boldsymbol{y}_2})\,\boldsymbol{x} = \boldsymbol{V}_{\boldsymbol{y}_1}\boldsymbol{x} + \boldsymbol{V}_{\boldsymbol{y}_2}\boldsymbol{x} = \mathcal{B}_{\mathrm{V}}(\boldsymbol{x}, \boldsymbol{y}_1) + \mathcal{B}_{\mathrm{V}}(\boldsymbol{x}, \boldsymbol{y}_2)$.

□

In contrast to circular convolution, VTB is neither commutative,

$$\mathcal{B}_{\mathrm{V}}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{V}_y\boldsymbol{x} \neq \boldsymbol{V}_x\boldsymbol{y} = \mathcal{B}_{\mathrm{V}}(\boldsymbol{y}, \boldsymbol{x}), \tag{2.21}$$

nor associative,

$$\mathcal{B}_{\mathrm{V}}(\boldsymbol{x}, \mathcal{B}_{\mathrm{V}}(\boldsymbol{y}, \boldsymbol{z})) = \boldsymbol{V}_{\boldsymbol{V}_z\boldsymbol{y}}\boldsymbol{x} \neq \boldsymbol{V}_z\boldsymbol{V}_y\boldsymbol{x} = \mathcal{B}_{\mathrm{V}}(\mathcal{B}_{\mathrm{V}}(\boldsymbol{x}, \boldsymbol{y}), \boldsymbol{z}). \tag{2.22}$$

Thus, a separate unbinding step is required for each binding, whereas circular convolution permits undoing multiple bindings in one step. It is, however, possible to flip the operands of a VTB binding in the bound state by multiplying with the permutation matrix,[2]

$$[\boldsymbol{V}_{\leftrightarrow}]_{ij} = \begin{cases} 1 & j = 1 + \left\lfloor \frac{i-1}{d'} \right\rfloor + d'[(i-1) \bmod d'], \\ 0 & \text{otherwise} \end{cases} \tag{2.23}$$

such that $\mathcal{B}_{\mathrm{V}}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{V}_{\leftrightarrow}\mathcal{B}_{\mathrm{V}}(\boldsymbol{y}, \boldsymbol{x})$. As an example, for $d = 4$, this matrix is

$$\boldsymbol{V}_{\leftrightarrow}^{(4)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The identity vector for VTB can be derived as the vector resulting in $\boldsymbol{V}_{\boldsymbol{i}_{\mathrm{V}}} = \boldsymbol{I}$:

Corollary 2 *(VTB identity vector). The identity vector for VTB is given by*

$$[\boldsymbol{i}_{\mathrm{V}}]_i = \begin{cases} d^{-\frac{1}{4}} & i \in \{(k-1)d' + k : k \leq d', k \in \mathbb{N}_{>0}\} \\ 0 & \textit{otherwise} \end{cases}. \tag{2.24}$$

Proof. By writing $\boldsymbol{i}_{\mathrm{V}}$ as $\boldsymbol{V}_{\boldsymbol{i}_{\mathrm{V}}}$, one can easily verify that $\boldsymbol{V}'_{\boldsymbol{i}_{\mathrm{V}}} = \boldsymbol{I} \Rightarrow \boldsymbol{V}_{\boldsymbol{i}_{\mathrm{V}}} = \boldsymbol{I}$. □

All vectors $\boldsymbol{u}$ that result in a perfectly orthogonal matrix $\boldsymbol{V}_{\boldsymbol{u}}$ are unitary because an orthogonal matrix is also unitary. Absorbing elements are required to fulfill $\boldsymbol{V}_{\boldsymbol{n}_{\mathrm{V}}}\boldsymbol{x} = c\boldsymbol{n}_{\mathrm{V}}$, which is not possible in general except for the trivial solution of the zero vector $\boldsymbol{n}_{\mathrm{V}} = (0, \ldots, 0)^{\top}$ as the solution for $\boldsymbol{n}_{\mathrm{V}}$ depends on $\boldsymbol{x}$.

3 Structured Representations

The main reason that VSAs are used in cognitive modeling is that the binding and superposition operations can be used to build up structured representations. For example, let us describe a scene containing a red square and a blue circle. Given almost orthogonal vectors *red*, *blue*, *square*, and *circle*, one possible way to encode this scene would be

[2] This is unrelated to the use of permutations for hiding in the MAP (multiply, add, permute) coding scheme by Gayler (2004). Hiding happens in VTB as part of the binding operator.

$$\boldsymbol{s} = \mathcal{B}(red, square) + \mathcal{B}(blue, circle). \tag{3.1}$$

The color of the square could then be recovered as

$$\begin{aligned}\mathcal{B}^+(\boldsymbol{s}, square) &= \mathcal{B}^+(\mathcal{B}(red, square), square) + \mathcal{B}^+(\mathcal{B}(blue, circle), square)\\ &\approx red + \text{noise}.\end{aligned} \tag{3.2}$$

Another viable approach to structure the representation would be to bind the object properties to tags like *color* and *shape*, and each individual object representation to a position tag or identifier:

$$\begin{aligned}\boldsymbol{o}_1 &= \mathcal{B}(red, color) + \mathcal{B}(square, shape),\\ \boldsymbol{o}_2 &= \mathcal{B}(blue, color) + \mathcal{B}(circle, shape),\\ \boldsymbol{s} &= \mathcal{B}(\boldsymbol{o}_1, obj_1) + \mathcal{B}(\boldsymbol{o}_2, obj_2).\end{aligned} \tag{3.3}$$

To retrieve the color of a specific shape in this representation scheme, each object needs to be retrieved; then the shape of the object needs to be compared to the target shape, and finally the color has to be unbound from the object with the matching shape.

3.1 Encoding Methods. Both examples of structured representations rely on pairwise binding of colors and shapes or of attributes to tags. We define this general method of encoding multiple pieces of information into a single vector to be *encoded with binding*.

Definition 6 *(encoding with binding). Given k pairs $(\boldsymbol{x}_i, \boldsymbol{y}_i) \in \mathbb{R}^d \times \mathbb{R}^d$, the encoding of these pairs into a single vector $\boldsymbol{m}$ with binding is given by*

$$\boldsymbol{m} = \sum_{i=1}^{k} \mathcal{B}(\boldsymbol{x}_i, \boldsymbol{y}_i). \tag{3.4}$$

An $\boldsymbol{x}_i$ can be recalled from such a trace as $\boldsymbol{x}_i \approx \hat{\boldsymbol{x}}_i = \mathcal{B}^+(\boldsymbol{m}, \boldsymbol{y}_i)$.

Recchia et al. (2015) proposed a different way to encode these pairwise relationships that we call *encoding with tagging*.

Definition 7 *(encoding with tagging). Given k pairs $(\boldsymbol{x}_i, \boldsymbol{y}_i) \in \mathbb{R}^d \times \mathbb{R}^d$ and a matrix $\boldsymbol{M} \in \mathbb{R}^{d\times d}$ with an approximate inverse $\boldsymbol{M}^+$ satisfying $\boldsymbol{M}^+\boldsymbol{M} \approx \boldsymbol{I}$, the encoding into a single vector with tagging is given by*

$$\boldsymbol{m} = \sum_{i=1}^{k} \boldsymbol{M}^{2i-1}(\boldsymbol{y}_i + \boldsymbol{M}\boldsymbol{x}_i) = \sum_{i=1}^{k} \boldsymbol{M}^{2i-1}\boldsymbol{y}_i + \boldsymbol{M}^{2i}\boldsymbol{x}_i. \tag{3.5}$$

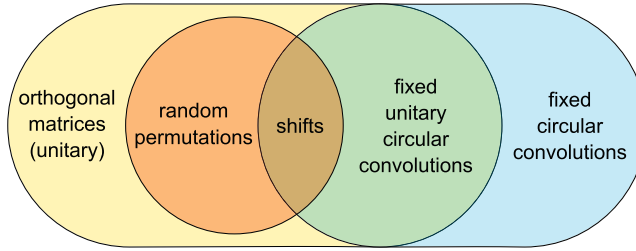

Figure 1. Venn diagram of the relationship between different choices of tagging matrices.

The retrieval of an $\boldsymbol{x}_i \approx \hat{\boldsymbol{x}}_i$ is accomplished with

$$\hat{\boldsymbol{x}}_i := (\boldsymbol{M}^{2c})^+ \boldsymbol{m}, \tag{3.6}$$

$$c = \arg\max_{j \in [1,\, k]} s(\boldsymbol{y}_i, (\boldsymbol{M}^{2j-1})^+ \boldsymbol{m}). \tag{3.7}$$

Recchia et al. (2015) proposed permutation matrices for $\boldsymbol{M}$, but a number of other choices are viable as well:

- For both presented binding operations, circular convolution and vector-derived transformation binding, the binding with a fixed vector can be expressed as a multiplication with a matrix $\boldsymbol{M}$. Thus, (repeated) binding to a fixed vector can be used for tagging.
- The matrix that shifts all vectors' elements by one (with wrap-around). This matrix is a special permutation matrix and at the same time represents a circular convolution binding to the fixed vector $(0, 1, 0, 0, \ldots)^\top$.
- Orthogonal (random) matrices. Note that orthogonal matrices (like permutation matrices or binding with a unitary fixed vector) have an exact inverse, which can be beneficial in unbinding.

All of these possible choices for $\boldsymbol{M}$ are related as shown in Figure 1.

4 Comparison of Binding and Encoding Methods

We now describe our empirical comparisons of the binding and encoding methods described. We consider aspects of the basic binding operation first, before moving on to measures of encoding capacity of lists and stacks. Finally, we consider the neural resources required when implementing the binding operations with the Neural Engineering Framework (Eliasmith & Anderson, 2003). (The relevant code and some precomputed data for these analysis can be found at https://github.com/ctn-archive/vtb.)

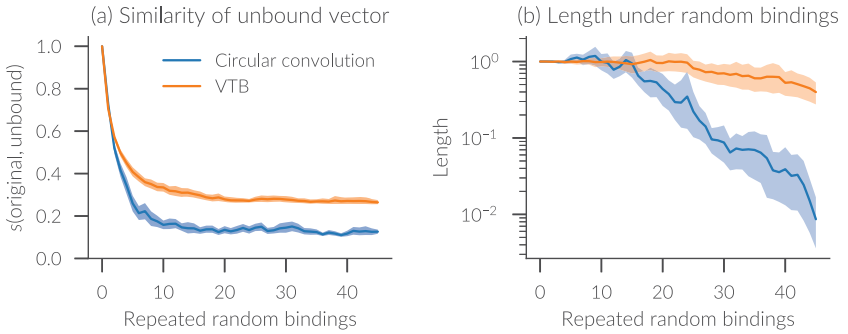

Figure 2. (a) Similarity of recovered vector after binding and unbinding with random vectors. (b) Change in vector norm with repeated binding of random vectors. Shaded areas represent bootstrapped 95% confidence intervals.

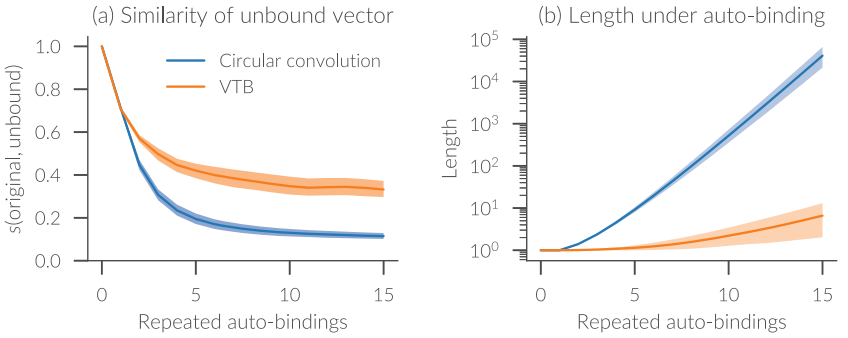

Figure 3. (a) Similarity of recovered vector after binding and unbinding with itself. (b) Change in vector norm with repeated binding to itself. Shaded areas represent bootstrapped 95% confidence intervals.

4.1 Binding Properties. For the binding operation, it is desired that after binding and unbinding, the resulting vector is similar to the original vector. Figure 2 shows how the similarity declines when binding and unbinding an increasing number of random vectors with circular convolution or VTB. While the similarity declines quickly for both binding methods, the VTB flattens out at a higher similarity. Furthermore, the change in the vector norm with increasing number of bindings is shown. Circular convolution quickly reduces the norm, which can be problematic in neural networks (see section 5), while the norm declines much more slowly with VTB.

In some situations, like encoding with tagging, the same vector is repeatedly bound to itself. This case has to be considered separately (see Figure 3). The results for the similarity after unbinding are comparable to the binding with random vectors. However, the vector norm will increase instead

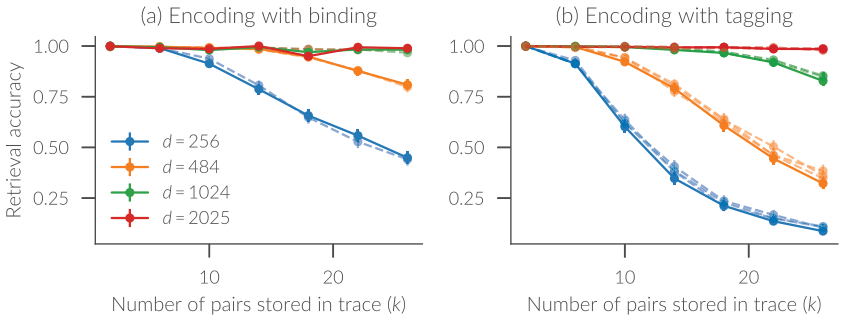

Figure 4. Retrieval accuracy from a list with encoding with binding (a) and encoding with tagging (b). Solid lines show results for VTB. Dashed lines show results for circular convolution. Results for unitary circular convolution matrices, shift matrices, and orthogonal matrices are shown, though they are similar enough that additional labeling is not specified. Error bars, mostly smaller than the marker size, represent 95% confidence intervals.

of decrease. This increase is much faster for circular convolution than for VTB.

4.2 List Encoding Capacity. To measure the list encoding capacity of the binding and encoding methods, we reproduced the first experiment from Recchia et al. (2015). A set of 1000 vectors with normally distributed components $x_i \sim \mathcal{N}(0, \sqrt{1/d})$ is created with $\mathbb{E}[\|\boldsymbol{x}\|] = 1$. From this set, 500 pairs are sampled with replacement, and k of these pairs are encoded into one vector. It is then tested whether the $\boldsymbol{x}$ of a pair $(\boldsymbol{x}, \boldsymbol{y})$ can be retrieved successfully given $\boldsymbol{y}$. This is the case if the unbound $\hat{\boldsymbol{x}}$ is more similar to $\boldsymbol{x}$ than any other vector in the initial set of 1000 vectors. For each data point, 1000 trials were averaged. The experiment was performed for dimensionalities $d \in \{256, 484, 1024, 2025\}$, adhering to the VTB constraint of square d. These dimensionalities are not exactly the same as used by Recchia et al. (2015), but they cover the same range.

Figure 4 shows the result for encoding with binding and tagging. The retrieval accuracy declines as the number of pairs encoded in the trace grows. Increasing the dimensionality also increases the storage capacity. Encoding with binding allows the storage of about twice as many pairs compared to encoding with tagging. Furthermore, there was virtually no difference in performance between different orthogonal matrices (like circular convolution or shift matrices) for the encoding with tagging. Circular convolution and VTB perform about the same with both encoding methods.

We repeated the experiment with nonorthogonal matrices—circular convolutions and VTB with a fixed, nonunitary vector (see Figure 5). A fixed nonorthogonal circular convolution matrix allows storing two pairs at most.

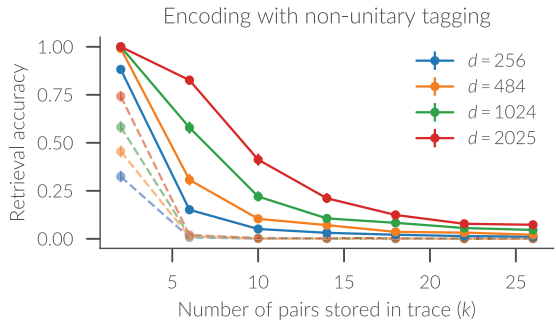

Figure 5. Retrieval accuracy for encoding with tagging with nonunitary VTB (solid lines) and circular convolution (dashed lines) matrices. Error bars, mostly smaller than the marker size, represent 95% confidence intervals.

Better retrieval accuracies are achieved with the fixed nonorthogonal VTB matrices, but the performance declines much faster for an increased number of pairs than with orthogonal tagging matrices.

4.3 Stack Encoding Capacity. The list (or pairwise) encoding capacity informs us only about the performance of the binding operations for flat representations. Now we look at the performance for encoding a stack with repeated bindings. The encoding of n vectors $\boldsymbol{x}_i$ into a stack is given by

$$\begin{aligned} \boldsymbol{m}_1 &= \boldsymbol{x}_1, \\ \boldsymbol{m}_{i+1} &= \mathcal{B}(\boldsymbol{m}_i, \boldsymbol{t}) + \boldsymbol{x}_{i+1}, \end{aligned} \tag{4.1}$$

where $\boldsymbol{t}$ is a random, fixed vector. As a result, for an n-item stack, the binding operator is applied $n-1$ times. To decode an item $\boldsymbol{x}_j$ at depth $\rho = n - j$ from the top, ρ unbindings of $\boldsymbol{t}$ have to be performed.

To test the encoding capacity, we performed 50 trials for each possible combination of number of items n and decoding depth ρ ($\rho < n$). The vectors x_i and t were chosen such that no pairwise similarity exceeded a threshold of 0.1. The dimensionality was set to 2025, but analogous results are obtained with other dimensionalities. Figure 6 shows the results. The more items are encoded in the stack, the less similar the retrieved items become to the original item. Note that this is also true for the top of the stack that one might expect to be unaffected by the depth of the stack. Nevertheless, decoded items from closer to the top of the stack will be more similar to the original item. Furthermore, we can observe that VTB performs considerably better than circular convolution.

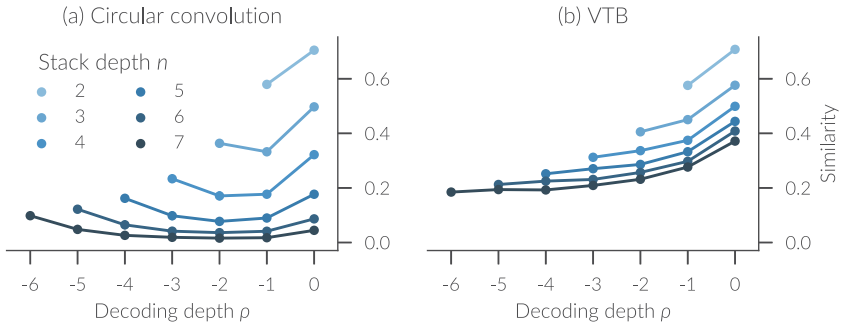

Figure 6. Similarity of retrieved vector from an encoded stack with (a) circular convolution and (b) VTB.

4.4 Neural Scaling. Apart from the mathematical performance, the number of neural resources to implement the binding operations should be considered when using them for cognitive modeling. We base our calculations on the assumption that the Neural Engineering Framework (NEF; Eliasmith & Anderson 2003) is used for such an implementation. The NEF provides a general method to implement given mathematical operations in a spiking neural network. This allows us to derive fair comparisons of required neural resources analytically, which is difficult with more traditional neural network approaches. Also, there are considerable numbers of NEF models (Eliasmith et al., 2012; Gosmann & Eliasmith, 2015; Crawford et al., 2016; Stewart & Eliasmith 2011) that make use of the Semantic Pointer Architecture (SPA; Eliasmith 2013). The SPA is an approach for implementing cognitive models with neural networks based on the NEF and vector symbolic architectures. In particular, it draws heavily on holographic reduced representations (Plate, 2003) with circular convolution as binding a operator. Thus, the results in this section are immediately applicable to a large number of models.

The circular convolution operation is most efficiently implemented in the Fourier space. Thus, both input vectors are projected into the Fourier space by a linear transform given by the discrete Fourier transform (DFT) matrix. For a d-dimensional vector, d complex Fourier coefficients are produced, but half of them are the complex conjugate of the other half. Thus, we only need to consider $d/2$ coefficients for either vector, which are multiplied together in pairs. Each complex multiplication can be expressed by four real valued multiplications, and thus a total of $2d$ multiplications is required. The number of neural resources required when implementing these with the NEF is proportional to the number of multiplies (Gosmann, 2015). Note that the DFT transform requires all-to-all connectivity, but the DFT matrix can be factorized to reduce the connectivity at the cost of additional layers.

Table 1: Maximum Similarity Constraints for Vectors Given Stack Depth and Binding Method to Ensure That the Unbound Items Are Most Similar to the Original Item in 95% of Cases.

Stack Depth n	$s_{\max,\circledast}$	$s_{\max,VTB}$
2	0.558	0.555
3	0.292	0.377
4	0.114	0.275
5	0.034	0.216

To implement the VTB, $d^{1/2}$ multiplications of $d^{1/2} \times d^{1/2}$ matrices with a vector are required. This results in a total of $d^{3/2}$ multiplications. One might notice that each column vector in $\boldsymbol{V_y}$ is scaled with the same component of the multiplied vector. This allows the encoding of all components in such a column into one NEF ensemble, together with the corresponding vector component. This requires only d ensembles to decode from, but each ensemble has to represent $d^{1/2} + 1$ dimensions. Accordingly the number of neurons in each ensemble has to be increased. It can be shown that for spiking LIF neurons in the NEF, the number of neurons needs to be scaled by $(d^{1/2} + 1)^{3/2} \approx d^{3/4}$ to keep the noise error constant (Gosmann, 2018). Summing over all ensembles, the neural resources have to be scaled by $d^{7/4}$, which is worse than doing pairwise multiplications. Due to the block structure of $\boldsymbol{V_y}$, all-to-all connectivity can be avoided.

Given that VTB has a worse scaling of neural resources but allows better decoding for deep hierarchies, it is worth investigating this trade-off more closely. In particular, we will look at decoding the item resulting in the lowest similarity from an n-item stack. From the analysis in the previous section and Figure 6, we can obtain a maximum similarity constraint that must not be exceeded between any of the potential vectors that might have been encoded to ensure successful cleanup of the unbound vector. As the actual similarity of the unbound vector $\hat{\boldsymbol{v}}$ to the original vector $\boldsymbol{v}$ is stochastic, we arbitrarily require that 95% of unbound vectors will exceed the similarity constraint (under the assumption that the similarity values are normally distributed). It follows that the maximum similarity between all candidate vectors must not exceed $s_{\max} = \mu - 1.645 \cdot \sigma$, where μ and σ are the mean similarity and standard deviation of the similarity $s(\hat{\boldsymbol{v}}, \boldsymbol{v})$. The resulting values for circular convolution and VTB are given in Table 1.

Given the maximum similarity constraint, we can estimate the capacity $M(s_{\max}, d)$ of unit vectors that fit into a d-dimensional vector space while respecting the constraint. Determining the exact number is an unsolved problem related to sphere packing, but lower and upper bounds have been derived (Wyner, 1965). We base our estimate on the lower bound from Wyner (1965) but account for the fact that we can always fit at least d

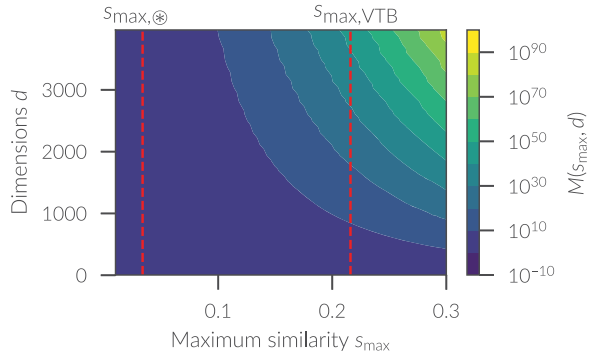

Figure 7. The lower bound for the maximum number of unit vectors that fit into a vector space without exceeding a pairwise-similarity of $s_{\max}$. The dashed red lines mark $s_{\max,\circledast}$ and $s_{\max,\mathrm{VTB}}$ for a stack depth of five.

orthogonal vectors into the vector space,

$$M(s_{\max}, d) = \max\left\{d, \frac{d}{d-1}\,\mathrm{B}\left(\frac{d+1}{2}, \frac{1}{2}\right)\left[\int_0^{\cos^{-1}s_{\max}} \sin^{d-2}\varphi \mathrm{d}\varphi\right]^{-1}\right\} \tag{4.2}$$

where $\mathrm{B}(x, y) = \int_0^1 t^{x-1}(1-t)^{y-1}\mathrm{d}t$ is the beta function. Figure 7 visualizes the function $M(s_{\max}, d)$. We justify using the lower bound compared to the upper bound because even achieving the lower bound is a hard problem.

As derived above, the vector dimensionality relates to the number of required multiplications, which can be taken as a measure of required neural resources for the binding. By converting the dimensionality to the number of multiplications, we can directly compare the capacity $M(s_{\max}, d)$ for the different binding methods as done in Figure 8. For a stack of depth of four (see Figure 8a), circular convolution allows for a larger number of vectors with respect to the similarity constraint, except for VTB with 121-dimensional vectors (1331 multiplications), though the difference is small.

However, this analysis does not account for the spiking noise of spiking neurons. Given $s_{\max,\circledast} < s_{\max,\mathrm{VTB}}$, less noise is acceptable if an equal performance is desired. In the NEF, the standard deviation of the spiking noise E decreases according to $E \propto 1/\sqrt{N}$, where N is the number of neurons. Thus, to achieve the same relative noise level, $s^2_{\max,\mathrm{VTB}}/s^2_{\max,\circledast}$ times more neurons are required when using circular convolution binding compared to VTB. When adjusting the number of multiplications to account for this, VTB performs better for any dimensionality.

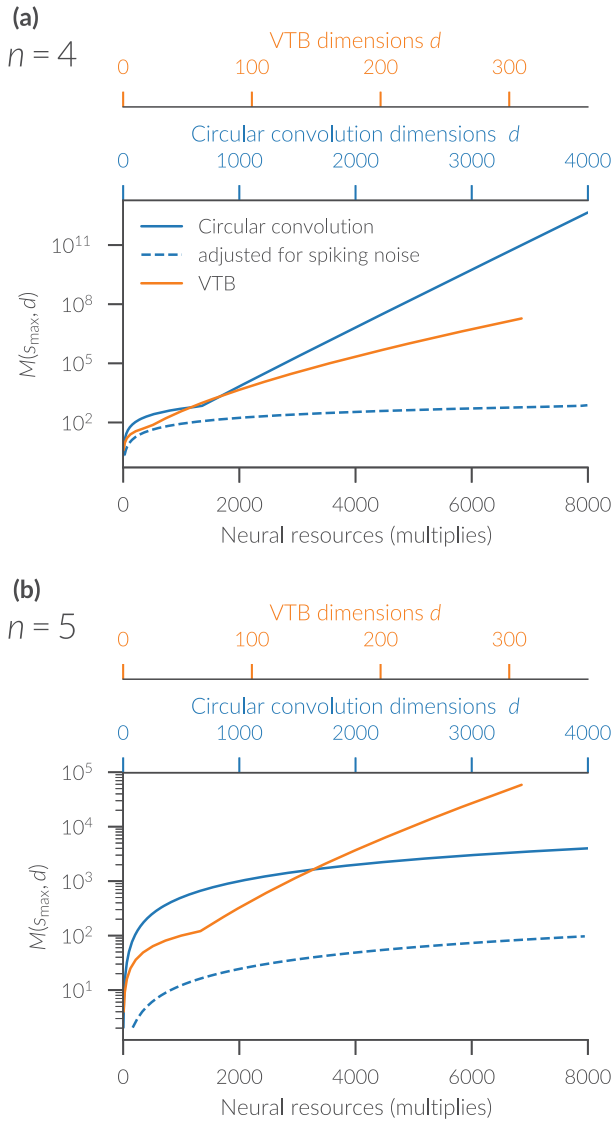

Figure 8. The capacity $M(s_{\max}, d)$ for different binding methods depends on the neural resources measured as the number of multiplications. The analysis in panel a is for a stack with three items, in panel b for a stack of four items. The top scales give the equivalent vector dimensionalities for a given number of multiplications. The dashed line results when adjusting the number of neural resources for circular convolution for the need of less spiking noise compared to VTB.

The same analysis performed for a stack of depth five (see Figure 8b) shows that up to 3270 multiplications, circular convolution allows for a larger number of vectors with respect to the similarity constraint. This corresponds to vectors with up to 1635 dimensions. Given additional neural resources to implement further multiplications, VTB allows for a larger number of vectors due to the looser similarity constraint. This corresponds to vector dimensions of 225 and up. Again, when adjusting for the neural spiking noise, VTB always gives a larger capacity.

For stacks with a depth exceeding five items, the similarity of the recovered vectors with circular convolution is too small to use this binding method. Finally, for stacks of depth two and three (no plots shown), the circular convolution binding allows for larger capacity even when adjusting for the neural noise.

5 Discussion

We presented a new binding method, vector-derived transformation binding, for use in vector-symbolic architectures. Compared to circular convolution, a commonly used binding method that underlies Plate's holographic reduced representations (Plate, 2003) and the Semantic Pointer Architecture (Eliasmith, 2013), it performs on par for flat structures. For such structures, we also found both of these binding methods to perform better with simple encoding with binding than with encoding with tagging.

This is in contradiction to the results of Recchia et al. (2015), who found encoding with tagging to perform better. We believe that our results are correct as they agree with the basic expectations that a superposition of twice as many vectors (as happens in the encoding with tagging) must lead to worse retrieval of individual elements. Note that our results for the encoding with tagging are in close quantitative agreement with the results from Recchia et al. (2015).

Because such flat structures are not always appropriate (e.g., in the n-back task, see Gosmann & Eliasmith, 2015, or for large-scale knowledge representation, see Crawford et al., 2016), we also tested the performance of the binding methods on a stack encoding. This covers the remaining of the two essential encoding schemes in a VSA. All other schemes reduce to either list or stack encoding (or a mixture of both) due to the distributivity of the binding operation. To create structure, elements need to be either bound to different "tags" and be superimposed (list encoding), or existing elements need to be "pushed down" by binding to a vector before adding a new vector.

We found the VTB to perform better for encoding stacks, especially for stacks with more than a few elements. This is due to two main effects. First, after repeated binding and unbinding, the resulting vector will be more similar to the original vector with VTB. Second, the vector norm will change less with each binding, facilitating a neural implementation

where the representational radius is limited. If the vector norm becomes too small, neural noise will destroy any remaining useful information, and if the vector norm becomes too large, neural saturation will distort the representation.

We believe that this improved stack encoding capacity could be beneficial in a number of scenarios. For example, it can reduce the number of required cleanup memories to remove noise when decoding from deep structures. Such deep structures could be useful for knowledge representation where information can be structured across multiple levels (e.g., a penguin is a bird is an animal). Furthermore, sequences are highly important in many tasks. While these could be encoded with a list encoding, this requires a known tag vector for each position, but a stack encoding requires only a single tag vector. Finally, VTB might prove useful for the representations of tree-like structures, such as parse trees in language processing. In addition to capturing the potential depth of these trees, VTB is noncommutative and thus retains which element is the left and right child of a tree node.

We were specifically interested in the requirement of neural resources when implementing the binding operations in a spiking neural network. While circular convolution requires only a linear scaling of neural resources, VTB requires a scaling by $O(d^{3/2})$. Even though this scaling is worse, it is offset in some instances by the better performance that requires fewer vector dimensions to get the same performance. In particular, this is the case for stacks of depth three or more when taking into account neural spiking noise.

Besides cognitive modeling, the NEF is used to program neuromorphic hardware (Mundy, Stewart, Terrence, & Furber, 2015; Knight, 2016). On such hardware, dense connectivity as required can be problematic (Mundy, 2016). While this can be circumvented by introducing additional layers for circular convolution, VTB requires less connectivity without additional layers (see Figure 9). Circular convolution requires 332,800 connections for binding two 64-dimensional vectors when using five neurons per dimension. VTB requires 5120 fewer connections, a total of 327,680, for the same task when representing the vectors as 16-dimensional subvectors (which is the default in the SPA). Note that VTB, however, uses many more postsynaptic neurons as more multiplications are required. When matching the number of multiplications or representing lower-dimensional subvectors, the connectivity advantage of VTB increases further.

Apart from these pure performance metrics, it should be noted that VTB is neither commutative nor associative, in contrast to circular convolution. This can have implications on possible encoding schemes for structure. For example, circular convolution does not allow the differentiation of the left and right operand due to commutativity, but VTB does. These differences can also have implications for cognitive modeling, as it might be necessary to undo each binding separately with VTB, but not with circular convolution due to the associativity properties. Note that this applies only to deep,

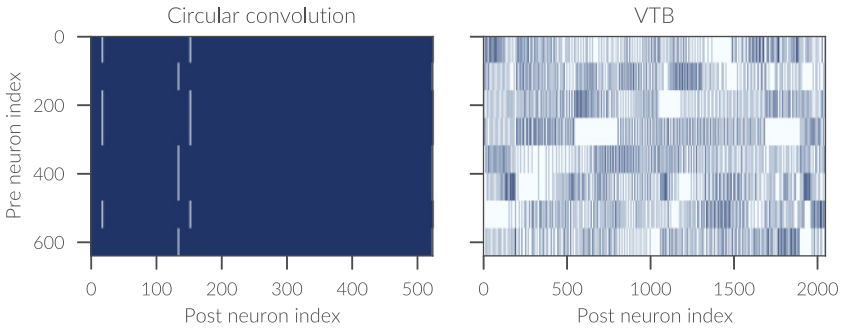

Figure 9. Neural connectivity for implementing circular convolution binding and VTB for 64-dimensional vectors with five neurons per dimension. Blue marks existing connections of any strength, while white marks nonexisting connections.

stack-like encodings. In a flat list encoding, the desired element can directly retrieved due to the distributivity of both operations (except if encoding with tagging is used, where an exhaustive search over all input vectors is required).

To summarize, VTB has promising properties that should be explored in future cognitive and neural network models. To facilitate research in this direction, we extended the freely available Nengo SPA Python library[3] that provides an implementation of the Semantic Pointer Architecture to allow the use of VTB instead of circular convolution.

Acknowledgments

This work has been supported by the Canada Research Chairs program, NSERC Discovery grant 261453, Air Force Office of Scientific Research grant FA8655-13-1-3084, CFI, and OIT.

References

Crawford, E., Gingerich, M., & Eliasmith, C. (2016). Biologically plausible, human-scale knowledge representation. *Cognitive Science*, *40*, 782–821.

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

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

Eliasmith, C., Stewart, T. C., Choo, X., Bekolay, T., DeWolf, T., Tang, Y., & Rasmussen, D. (2012). A large-scale model of the functioning brain. *Science*, *338*, 1202–1205.

[3] https://www.nengo.ai/nengo-spa/.

Gayler, R. W. (1998). Multiplicative binding, representation operators and analogy (Workshop Poster). In *Advances in analogy research: Integration of theory and data from the cognitive, computational, and neural sciences*. Sofia, Bulgaria: New Bulgarian University.

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

Gosmann, J. (2015). *Precise multiplications with the NEF*. Waterloo, ON: University of Waterloo.

Gosmann, J. (2018). *An integrated model of context, short-term, and long-term memory*. PhD diss., University of Waterloo.

Gosmann, J., & Eliasmith, C. (2015). A spiking neural model of the N-back task. In *Proceedings of the 37th Annual Meeting of the Cognitive Science Society* (pp. 812–817). Cognitive Science Society.

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

Knight, J., Voelker, A. R., Mundy, A., Eliasmith, C., & Furber, S. (2016). Efficient SpiNNaker simulation of a heteroassociative memory using the Neural Engineering Framework. In *Proceedings of the 2016 International Joint Conference on Neural Networks*. Piscataway, NJ: IEEE.

Mitchell, J., & Lapata, M. (2010). Composition in distributional models of semantics. *Cognitive Science*, *34*, 1388–1429.

Mundy, A. (2016). *Real time spaun on SpiNNaker: Functional brain simulation on a massively-parallel computer architecture*. PhD diss., University of Manchester.

Mundy, A., Stewart, J., Terrence, C., & Furber, S. (2015). An efficient SpiNNaker implementation of the Neural Engineering Framework. In *Proceedings of the 2015 International Joint Conference on Neural Networks*. Piscataway, NJ: IEEE.

Plate, T. A. (2003). *Holographic reduced representation: Distributed representation for cognitive structures*. Stanford, CA: CSLI Publications.

Recchia, G., Sahlgren, M., Kanerva, P., & Jones, M. N. (2015). Encoding sequential information in semantic space models: Comparing holographic reduced representation and random permutation. *Computational Intelligence and Neuroscience*, *2015*, 986574.

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

Stewart, T. C., & Eliasmith, C. (2011). Neural cognitive modelling: A biologically constrained spiking neuron model of the tower of Hanoi task. In *Proceedings of the 33rd Annual Meeting of the Cognitive Science Society* (pp. 656–661). Cognitive Science Society.

Wyner, A. D. (1965). Capabilities of bounded discrepancy decoding. *Bell System Technical Journal*, *44*, 1061–1122.

Received September 30, 2018; accepted December 18, 2018.