Connection Science
Vol. 23, No. 2, June 2011, 145–153

Taylor & Francis
Taylor & Francis Group

Neural representations of compositional structures: representing and manipulating vector spaces with spiking neurons

Terrence C. Stewart*, Trevor Bekolay and Chris Eliasmith

Centre for Theoretical Neuroscience, University of Waterloo, 200 University Avenue West, Waterloo, Ontario, Canada N2L 3G1

(*Received 6 March 2011; final version received 8 March 2011*)

This paper re-examines the question of localist *vs.* distributed neural representations using a biologically realistic framework based on the central notion of neurons having a *preferred direction vector*. A preferred direction vector captures the general observation that neurons fire most vigorously when the stimulus lies in a particular direction in a represented vector space. This framework has been successful in capturing a wide variety of detailed neural data, although here we focus on cognitive representation. In particular, we describe methods for constructing spiking networks that can represent and manipulate structured, symbol-like representations. In the context of such networks, neuron activities can seem both localist and distributed, depending on the space of inputs being considered. This analysis suggests that claims of a set of neurons being localist or distributed cannot be made sense of without specifying the particular stimulus set used to examine the neurons.

Keywords: neural engineering framework; preferred direction vectors; neural representation; distributed representation; localist representation; vector symbolic architectures

1. Neural representation

Simple scalar values of behaviourally relevant variables (such as head tilt) can be understood as being encoded in the brain as shown in Figure 1 (Eliasmith and Anderson 2003). This pattern is found in both visual and motor areas, including head orientation detection and eye position control. Each line is the *tuning curve* for a different neuron, showing how its firing rate changes as the value being represented ($\boldsymbol{x}$) changes. We can characterise this phenomenon by noting that the neurons fall into two groups, sometimes referred to as *on* neurons and *off* neurons. The on neurons 'prefer' (i.e. they fire more quickly for) positive values of $\boldsymbol{x}$ (tilting the head forward; looking to the right) and the off neurons prefer negative values (tilting backward; looking left).

We can capture this effect mathematically by noting that the current flowing into on-neurons is proportional to x, and for off-neurons, it is proportional to $-x$. Indeed, a close match to empirical data is found if we assume that the total current J flowing into each neuron is $J = \alpha px + J_b$, where x is the scalar value to be encoded, p is the neuron's preferred value (1 for on neurons

*Corresponding author. Email: tcstewar@uwaterloo.ca

ISSN 0954-0091 print/ISSN 1360-0494 online

DOI: 10.1080/09540091.2011.571761
http://www.informaworld.com

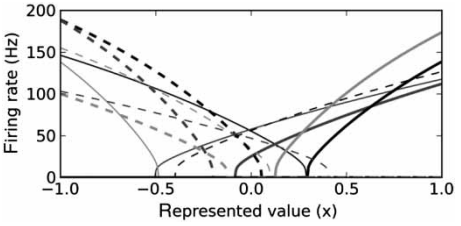

Figure 1. Typical tuning curves for neural representation. Each line shows the change in firing for a single neuron as the value being represented goes from −1 to 1.

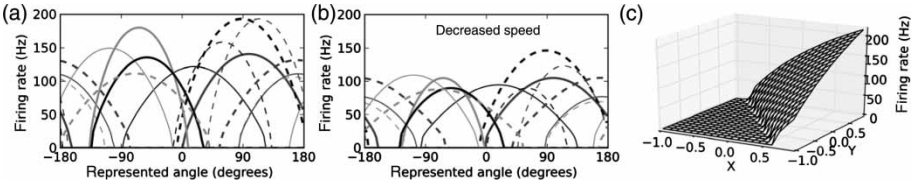

Figure 2. Tuning curves for neurons representing two dimensions (direction and speed). (a) Shows typical tuning curves for 12 neurons for different angles. (b) Shows the same neurons, but with a different speed. (c) Shows the full two-dimensional tuning curve for one particular neuron over all combinations of speed and direction.

and −1 for off neurons), α is a neuron-specific scaling factor, and J_b is the background current.[1] Varying p, α, and J_b produce all of the different tuning curves seen in Figure 1.

However, this is not the only sort of tuning curve found in the brain. Famously, Georgopolous, Schwartz, and Kettner (1986) found tuning curves in monkey motor cortex as in Figure 2(a), where the firing of each neuron is dependent on the direction of motion of the monkey's arm, and each neuron has a preferred angle. While they characterise this curve as a cosine, we have shown (Eliasmith and Anderson 2003) that it is more useful to take the same approach as in Figure 1 by allowing $\boldsymbol{p}$ to be a *preferred direction vector*, a two-dimensional vector chosen randomly from the unit circle. Furthermore, we assume that the neurons are not representing just the direction of motion; rather, they represent both the direction and magnitude of the arm movement (i.e. a two-dimensional, rather than one-dimensional, space). This provides a better match to empirical data (Todorov 2000) by having the input current be calculated as in Equation (1), with $\boldsymbol{p} \cdot \boldsymbol{x}$ being the dot product between the preferred direction vector and the movement speed and direction. This approach to neural representation generalises to any number of dimensions, including the single-dimensional case from Figure 1.

$$J = \alpha \boldsymbol{p} \cdot \boldsymbol{x} + J_b. \tag{1}$$

One key consequence is that tuning curves are dependent on the particular range of stimuli chosen. For example, Figure 2(a) and (b) show the tuning curves (as the direction of motion varies) for exactly the same neurons. The only difference between the figures is the *speed* of motion. Changing the speed changes the tuning curves, as was observed in later studies of this area (Todorov 2000). The full tuning curve can be seen in Figure 2(c) as a three-dimensional plot.

2. Decoding representations

The technique of defining a preferred direction vector for each neuron in a population provides a general-purpose method for encoding any vector of numerical values using the spiking activities of a group of neurons. To complement this, we also need to be able to *decode* these representations;

that is, given the spike trains of a set of neurons, we must be able to determine what value $\boldsymbol{x}$ was encoded. This is crucial because it places limits on what information is available to downstream neurons. Information that cannot, in principle, be decoded cannot be used.

One method for doing this is to determine *decoding* vectors $\boldsymbol{d}$ for each neuron. If we take the output of each neuron in the population that encodes $\boldsymbol{x}$, multiply by that neuron's decoding vector $\boldsymbol{d}$, and add the results across all neurons, we get an estimate of $\boldsymbol{x}$. As we have shown (Eliasmith and Anderson 2003), this is a standard least-squared-error minimisation problem, and $\boldsymbol{d}$ can be calculated via Equation (2), where a_i is the firing rate of neuron i for a particular value of $\boldsymbol{x}$, and the integration is the over all values of $\boldsymbol{x}$.

$$\boldsymbol{d} = \Gamma^{-1}\Upsilon \quad \Gamma_{ij} = \int a_i a_j \mathrm{d}x \quad \Upsilon_j = \int a_j \boldsymbol{x} \mathrm{d}x \tag{2}$$

This fixed set of decoding vectors $\boldsymbol{d}$ allows for the decoding of any vector $\boldsymbol{x}$ that can be encoded by the population of neurons.

3. Computing with connection weights

Given this method for encoding a vector value in a group of neurons, and decoding that value from the resulting activity of those neurons, we can turn to the question of *how* these neurons come to fire in this way. That is, what is the source of the input current we find with Equation (1)? For the vast majority of neurons, the main input is from synaptic connections with other neurons. In order to understand how a group of neurons comes to represent a particular value, we need to find synaptic connection weights from previous neurons that will cause them to produce the observed tuning curves.

One method for finding these connection weights is to learn them by providing an error signal. We can start with random connection weights ω_{ij} and then provide a feedback signal $\boldsymbol{R}$ which is the difference between the represented value and the desired value. The synaptic connection strength can then be updated according to Equation (3), where J_{ij} is the amount of current presently flowing from neuron i to neuron j, and κ is a learning rate constant. This is a biologically plausible local Hebbian learning rule, and is sufficient for learning complex functions such as integration and sophisticated nonlinearities (Bekolay 2010). For example, Figure 3 shows that this rule is sufficient to learn multiplication in 30 s (simulated time; approximately one hour of computer time is needed on modern processors).

$$\Delta\omega_{ij} = \kappa J_{ij} \alpha_j \boldsymbol{p} \cdot \boldsymbol{R}. \tag{3}$$

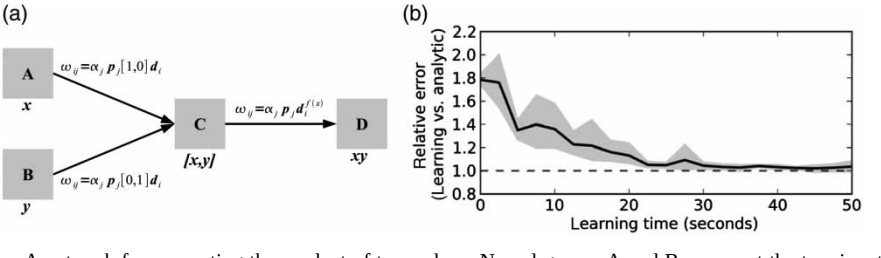

Figure 3. A network for computing the product of two values. Neural groups A and B represent the two inputs (with tuning curves as in Figure 1). Neural group C represents both values at once (with tuning curves as in Figure 2). Neural group D represents the product of the two inputs (tuning curves as in Figure 1, as shown in left). Connection weights can be learned via Equation (3) as shown on the right (grey area is 95% bootstrap confidence interval over 10 simulation runs), or computed analytically by evaluating the equations shown on the arrows.

However, if we want to develop large, complex models with hundreds of thousands of neurons, such a learning rule may be too slow to be practical from the standpoint of numerical simulations, requiring hundreds of hours of computer time. Instead, we can also directly analytically compute the connection weights that will compute the desired function. For any linear function $\boldsymbol{y} = \boldsymbol{Mx}$, this can be done with Equation (4), where d_i is the optimal decoder for neuron i found using Equation (2). For nonlinear functions $\boldsymbol{y} = f(\boldsymbol{x})$, we also use Equation (4), but compute a different decoder $\boldsymbol{d}^{f(x)}$ with Equation (5). The result is functionally equivalent to the weights found via the computationally much more expensive learning rule, and forms the basis of the Neural Engineering Framework (Eliasmith and Anderson 2003; Bekolay 2010).

$$\omega_{ij} = \alpha_j \boldsymbol{p}_j \boldsymbol{M} \boldsymbol{d}_i \tag{4}$$

$$\mathbf{d} f(x) = \Gamma^{-1} \Upsilon \quad \Gamma_{ij} = \int a_i a_j \, \mathrm{d}x \quad \Upsilon_j = \int a_j f(\boldsymbol{x}) \, \mathrm{d}x. \tag{5}$$

4. Tuning curves and computation

With the ability to connect spiking neurons so as to compute sophisticated functions, we now re-visit our previous notion of a 'tuning curve'. In particular, how should we characterise the tuning curve of the *output* neurons (group D) in Figure 3? To make the example concrete, we note that multiplication of values occurs in many models of vision, especially those involving visual attention. Group A neurons may represent the presence of a stimulus at a particular location, and group B may represent the amount of attention being focused on that location. If attention is held constant (at a value of 1), we can find a tuning curve for the neurons in group B by varying the strength of the stimulus (Figure 4(a)). However, if we now adjust the strength of attention (to 0.5 or -1), we find that the tuning curves have shifted (Figure 4(b)) or even reversed (Figure 4(c)). Such effects have been reported in the tuning curves of individual neurons recorded from monkeys (Womelsdorf, Anton-Erxleben, and Treue 2008).

The traditional way of thinking about this is to say that changing attention (group B) modulates the tuning curves over the stimulus (group A). However, it is just as valid to say that group A modulates the tuning curve from group B, given the symmetry of the connections. Another approach, similar to that shown in Figure 2(c), suggests that we can think of the neurons in group D as having a two-dimensional tuning curve that is sensitive to both the stimulus strength and the level of attention. However, this characterisation obscures the fact that there is a very precise pattern in this tuning curve in that the output neurons only represent the *product* of the values,

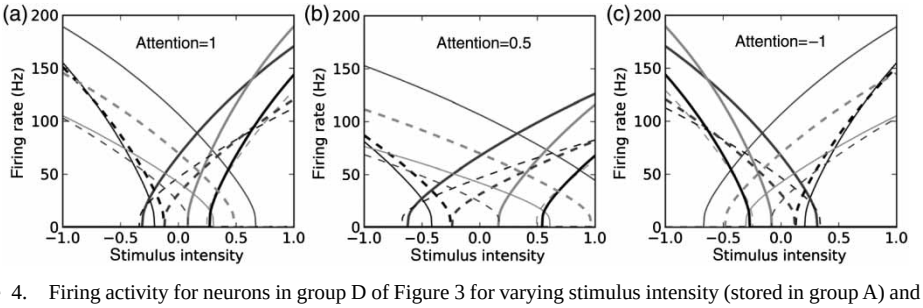

Figure 4. Firing activity for neurons in group D of Figure 3 for varying stimulus intensity (stored in group A) and level of attention (stored in group B). Each line shows exactly the same neuron in each plot; the only difference is the value stored in group B. It thus appears as though adjusting attention modulates the tuning curves of each neuron. Alternatively, all these neurons have *fixed* unchanging tuning curves that are responsive to the *product* of stimulus intensity and attention.

and so they will fire with exactly the same pattern for an input of A = 0.5, B = 1, and A = 1, B = 0.5.

We recommend a third approach to analysing group D. The core idea is that this group of neurons represents a *computed value*. This means the tuning curves do not change, and neither does the function being computed. Instead, the neurons are simply representing a value that is *computed from* whatever values are represented in the previous neurons (groups A and B). It is this value that best summarises the behaviour of these neurons. In this particular case, we can see that the firing behaviour of the neurons in group D is most compactly described as encoding the product of stimulus strength and attention, rather than encoding stimulus strength using tuning curves that are modulated by attention. In general, we suggest that we can think of all tuning curves as (usually quite simple) relationships between the value being represented and the firing rates of the neurons, rather than the (potentially highly complex) relationship between the various different inputs and the firing rates.

5. Vectors and language-like structures

Since we can find neural connection weights that compute particular functions, we are in a position to form neural implementations of highly complex systems, as long as those systems can be expressed using vectors and functions on those vectors. There is a family of such systems capable of representing and manipulating language-like structures: vector symbolic architectures (VSAs; for a summary, see Gayler 2003, and for more details, see Kanerva 2009 and Plate 2003). In these approaches, each basic symbol (such as RED or COLOUR) corresponds to one high-dimensional vector (either chosen randomly or based on semantic similarity to other symbols). Structured information can be formed by combining vectors in two ways: superposition (+), which creates a new vector that is similar to the original vectors and binding (⊗), which produces a vector that is dissimilar to the originals. For example, the vector for representing a red square could[2] be computed as COLOUR ⊗ RED + SHAPE ⊗ SQUARE.

For this paper, we use a particular VSA known as Holographic Reduced Representations (HRRs; Plate 2003), where superposition is done by addition and binding is done by circular convolution. All basic vectors are randomly chosen 100-dimensional vectors of unit magnitude. Figure 5 shows a network capable of binding two vectors together (i.e. computing A ⊗ B). This network with fixed connection weights computes the circular convolution of whatever two input vectors are provided. While Figure 5 shows that these connection weights can be learned, the learning process is slower

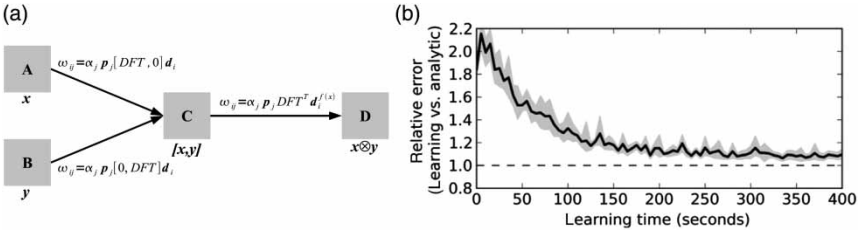

Figure 5. A network for computing the circular convolution of two vectors. Neural groups A and B represent the 100-dimensional vectors $\boldsymbol{x}$ and $\boldsymbol{y}$. Neural group C represents the Fourier transform of these two vectors. Neural group D stores the resulting 100-dimensional vector. Circular convolution is identical to multiplication after a Fourier transform. As the vectors are transferred from A and B–C, they are multiplied by the discrete Fourier transform (DFT) matrix, producing two new vectors that are stored in C. For the connections to D, we find decoders $\boldsymbol{d}^{f(x)}$ that compute the element-wise product of these vectors and multiply by the inverse DFT to arrive at the final result, as shown in left. The synaptic connection weights can also be learned, as is shown on the right for three-dimensional convolution (grey area is 95% bootstrap confidence interval).

as the number of dimensions increases. Consequently, the models in the rest of this paper use connection weights analytically derived using using Equations (4) and (5).

6. Tuning curves for vector-based symbols

In previous work, we have used neural models of VSAs constructed in this manner to explain list memory (Choo and Eliasmith 2010), context dependence in the Wason card-flipping task (Eliasmith 2005), planning and goal memory in the Tower of Hanoi (Stewart and Eliasmith 2011), and generalisation with rule induction in Raven's Matrices (Rasmussen and Eliasmith 2010). In this paper, rather than demonstrating a new model of this form, we examine a fundamental characteristic of all these models: the behaviour of individual neurons as they switch from representing one symbol structure to another.

Importantly, a group of neurons can represent *any* vector. The same neurons can represent the vector for RED or the vector for BLUE by changing their firing patterns, just as the neurons in Figure 1 can represent any value from -1 to 1 by changing their firing patterns. This means the same neurons can also represent SQUARE or CIRCLE, or even SQUARE + CIRCLE. Indeed, these neurons could also represent COLOUR $\otimes$ RED + SHAPE $\otimes$ SQUARE or COLOUR $\otimes$ BLUE + SHAPE $\otimes$ CIRCLE, or even SUBJECT $\otimes$ SALLY + ACTION $\otimes$ THROWS + OBJECT $\otimes$ BALL.[3]

The actual tuning curves for these neurons can be derived from Equation (1). However, since $\boldsymbol{x}$ is a 100-dimensional vector, this means that showing a full tuning curve would be like Figure 2(c), but 100-dimensional. Since this cannot be depicted, we instead show the change in the firing rate as the population is driven to represent a small selected group of inputs. The vectors for each of these inputs are randomly chosen 100-dimensional vectors, as are the preferred direction vectors for each neuron. Figure 6(a)–(c) show the tuning curves for *exactly the same 12 neurons* as they represent different sets of vectors. Each of these tuning curves is a small subsection of the full 100-dimensional tuning curve.

We first note that some neurons seem to use a localist encoding. For example, in 6(a), some neurons only fire for SQUARE, some for CIRCLE, and so on. However, other neurons look much more distributed, gradually changing value and not uniquely picking out any particular symbol. Importantly, since the current flowing into each neuron is well-approximated by Equation (1), the only difference between these neurons must be due to the parameters $\boldsymbol{p}$ (the preferred direction vector), α (scaling), and J_b (background current). A neuron that seems to be selective for CIRCLE must then have a preferred direction vector $\boldsymbol{p}$ that is similar to the vector for CIRCLE.

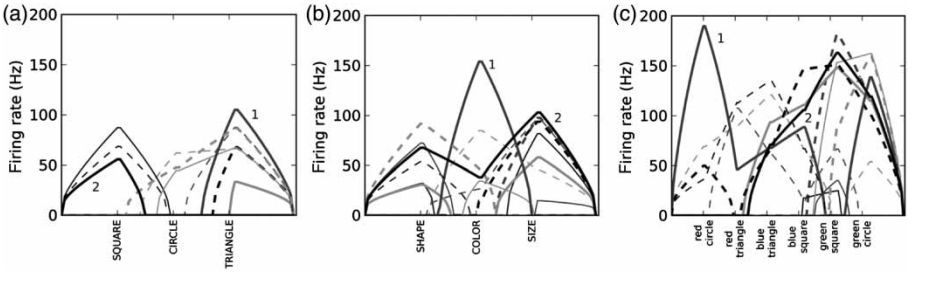

Figure 6. Firing activity for neurons representing 100-dimensional VSA vectors. Each graph shows the same 12 neurons, but varies the represented vectors over a different range. In (a), neuron 1 appears localist as it only fires for TRIANGLE, but in (b) we see that it also fires for COLOUR. In (c) we see the same neuron fires at different rates for many different coloured shapes (except for green squares). Similarly, neuron 2 only fires for SQUARE in (a), everything in (b), and everything except the red shapes in (c). Whether a neuron appears localist is thus dependent on the stimuli presented.

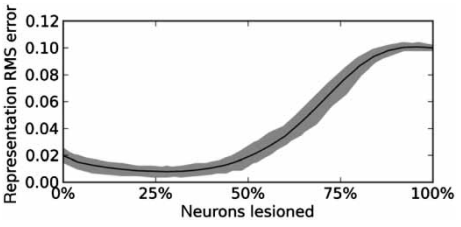

Figure 7. Representation accuracy as neurons are lesioned in order of locality. 5000 neurons are used to represent a 100-dimensional vector $\boldsymbol{x}$. Neurons are destroyed in order of sensitivity to the particular x value represented, starting with the most local. Grey area is minimum/maximum error over 50 trials. The most local 50% of neurons can be completely destroyed before we see significant decreases in representation accuracy.

Exactly the same reasoning applies to 6(b), where the neurons represent different categories (SHAPE, COLOUR, SIZE, etc.) and 6(c), where complex structures such as 'red triangle' (COLOUR $\otimes$ RED + SHAPE $\otimes$ TRIANGLE) are represented. Importantly, we see that the same neurons can appear to be localist or distributed, depending on what range of inputs are being examined. However, in all cases, we can accurately characterise their behaviour by thinking of them having a preferred direction vector, and that the group of neurons as a whole is representing a particular vector in a high-dimensional space.

7. Lesioning 'localist' neurons

Since it appears that some neurons do preferentially fire for particular symbols, one might ask how important those neurons are. That is, what happens to the neural group's ability to represent CIRCLE when the neurons with preferred direction vectors near CIRCLE are removed? We can evaluate this by gradually destroying neurons, in order of how close their $\boldsymbol{p}$ value is to the vector for CIRCLE. Figure 7 shows the accuracy of the representation as this occurs, up until all the neurons are destroyed. The representation is robust and is quite capable of representing the CIRCLE without any CIRCLE-specific neurons. This means that, even though we may find neurons that seem to only respond when representing particular items, these neurons are not actually required for accurate neural processing. The presence of other neurons with some response to this vector is enough to recover the information.

8. Conclusions

Given these observations, what can we conclude about localist and distributed representations? Inferences that move from examining the activity of a small set of neurons to claims about the general nature of neural representation come in various forms. One common inference is that, if a neuron fires to a specific stimulus only (e.g. a 'grandmother') it is a localist representation. With the notion of preferred direction vectors in hand, we can see that this is a poor inference in two respects.

First, seen in Figure 7, there is not necessarily a special dependence on system performance with such neurons. In contrast, models which employ localist representations typically have a special dependence on localist nodes, namely, if that node is destroyed, the corresponding representation is no longer available. This does not occur in the models presented here, even though the neurons can appear localist.

Second, these kinds of observations do not demonstrate that such neurons are, in any special sense, 'tuned' to a specific stimulus. After all, every neuron has a preferred direction vector to which its responses are most tightly tuned. That a neuron's preferred vector lines up with an item included in a stimulus set does not make it special; all neurons will line up best with *some* combination of stimulus features. Indeed, any experiment is unlikely to test all possible stimuli, especially as we consider neurons involved in the complexities of higher level cognition, rather than more peripheral neurons dealing with sensory and motor systems.

In short, such considerations suggest two things. First, neural systems are likely to have both apparently local and apparently non-local neurons, with respect to a particular set of stimuli. Second, individual neurons are likely to be both local and non-local with respect to different sets of stimuli and the experimenter will not, in general, know whether an entire space of stimuli has been probed. Thus, claims about the locality of a particular neural representation should be made only with respect to a set of stimuli. Perhaps more importantly, appearing localist does not guarantee any functional consequences, such as a lack of robustness to damage.

In the end, we would suggest that characterising neural representations in terms of preferred direction vectors in a vector space provides a more systematic and useful method for understanding neural representation. Arguments about whether or not neural representations are local or distributed in general (Bowers 2009) seem largely misguided. Instead, a focus on how individual neurons contribute to the representation and transformation of vector spaces can effectively guide our modelling efforts, from the level of simple one-dimensional representations found across the animal kingdom to the level of structured, language-like representation.

Acknowledgements

This research was supported by funding from the SHARCNET Research Fellowship Programme, the Natural Sciences and Engineering Research Council of Canada, the Canada Foundation for Innovation, the Ontario Innovation Trust, and the Canada Research Chairs program.

Notes

1. The nonlinearity seen in Figure 1 is dependent on the neural model used to convert current into spikes. For this paper, we use the standard Leaky Integrate-and-Fire neuron.
2. There are many different ways that this situation could be represented using VSAs, much as there are many different ways of representing data in computing (e.g. a linked list, an array, or a hash table). In each case, a specific representational approach is chosen because its characteristics are compatible with how it will be used, and we have used this approach in most of our models.
3. The limit on what a population of neurons can represent does not lie within that population, but rather in the range of decodings that can be accurately implemented by downstream neurons. This is usually determined by the capabilities of a cleanup memory and we have shown that these neural models scale up to the size of standard adult vocabulary (Stewart, Tang, and Eliasmith 2011).

References

Bekolay, T. (2010), 'Learning Nonlinear Functions on Vectors: Examples and Predictions', Technical Report CTN-TR-20101217-010, University of Waterloo Centre for Theoretical Neuroscience.

Bowers, J. (2009), 'On the Biological Plausibility of Grandmother Cells: Implications for Neural Network Theories in Psychology and Neuroscience'. *Psychological Review*, 116(1), 220–251.

Choo, F., and Eliasmith, C. (2010), 'A Spiking Neuron Model of Serial-Order Recall', in *32nd Annual Conference of the Cognitive Science Society*, eds. R. Cattrambone and S. Ohlsson, Austin, TX: Cognitive Science Society, pp. 2188–2193.

Eliasmith, C. (2005), 'Cognition with Neurons: A Large-scale, Biologically Realistic Model of the Wason Task', in *27th Annual Conference of the Cognitive Science Society*, eds. G. Bara, L. Barsalou, and M. Bucciarelli, Austin, TX: Cognitive Science Society, pp. 624–629.

Eliasmith, C., and Anderson, C.H. (2003), *Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems*, Cambridge, MA: MIT Press.

Gayler, R. (2003), 'Vector Symbolic Architectures Answer Jackendoff's Challenges for Cognitive Neuroscience', in *International Conference on Cognitive Science*, ed. Peter Slezak, Sydney, Australia: University of New South Wales, pp. 133–138.

Georgopolous, A.P., Schwartz, A., and Kettner, R.E. (1986), 'Neuronal Population Coding of Movement Direction', *Science*, 260, 47–52.

Kanerva, P. (2009), 'Hyperdimensional Computing: An Introduction to Computing in Distributed Representation with High-Dimensional Random Vectors', *Cognitive Computation*, 1(2), 129–159.

Plate, T. (2003), *Holographic Reduced Representations*, Stanford, CA: CSLI.

Rasmussen, D., and Eliasmith, C. (2010), 'A Neural Model of Rule Generation in Inductive Reasoning', in *32nd Annual Conference of the Cognitive Science Society*, eds. R. Cattrambone, Austin, TX: Cognitive Science Society, pp. 61–66.

Stewart, T.C., and Eliasmith, C. (2011), 'Neural Cognitive Modelling: A Biologically Constrained Spiking Neuron model of the Tower of Hanoi Task', in *33rd Annual Meeting of the Cognitive Science Society*, eds. L. Carlson, C. Hölscher, and T. Shipley, Austin, TX: Cognitive Science Society.

Stewart, T.C., Tang, Y., and Eliasmith, C. (2011), 'A Biologically Realistic Cleanup Memory: Autoassociation in Spiking Neurons', *Cognitive Systems Research*, 12, 84–92.

Todorov, E. (2000), 'Direct Cortical Control of Muscle Activation in Voluntary Arm Movements: A Model', *Nature Neuroscience*, 3, 391–398.

Womelsdorf, T., Anton-Erxleben, K., and Treue, S. (2008), 'Receptive Field Shift and Shrinkage in Macaque Middle Temporal Area Through Attentional Gain Modulation', *Journal of Neuroscience*, 28(36), 8934–8944.