A Biologically Realistic Cleanup Memory: Autoassociation in Spiking Neurons

Terrence C. Stewart (tcstewar@uwaterloo.ca)
Yichuan Tang (y3tang@uwaterloo.ca)
Chris Eliasmith (celiasmith@uwaterloo.ca)
Centre for Theoretical Neuroscience, University of Waterloo
Waterloo, ON, N2L 3G1

Abstract

Methods for cleaning up (or recognizing) states of a neural network are crucial for the functioning of many neural cognitive models. For example, Vector Symbolic Architectures provide a method for manipulating symbols using a fixed-length vector representation. To recognize the result of these manipulations, a method for cleaning up the resulting noisy representation is needed, as this noise increases with the number of symbols being combined. While these manipulations have previously been modelled with biologically plausible neurons, this paper presents the first spiking neuron model of the cleanup process. We demonstrate that it approaches ideal performance and that the neural requirements scale linearly with the number of distinct symbols in the system. While this result is relevant for any biological model requiring cleanup, it is crucial for VSAs, as it completes the set of neural mechanisms needed to provide a full neural implementation of symbolic reasoning.

Keywords: autoassociative memory; neural engineering framework; vector symbolic architectures; holographic reduced representation

Autoassociative Memory

A fundamental component of many cognitive architectures is an autoassociative memory. This is a system that can be provided with a partial or noisy version of a previously stored memory and will in turn provide a complete and more accurate version of that memory. This can be seen in ACT-R's declarative memory system (Anderson & Lebiere, 1998), CLARION's non-action-centered subsystem (Sun, 2006), RAAM's compressor and reconstructor (Pollack, 1988), and many other cognitive models. This capability can be implemented using a wide variety of approaches, including multilayer perceptrons, Hopfield networks, and any prototype-based classifier.

The particular use of autoassociative memory of importance to this paper is as a *cleanup memory* for cognitive operations. Recent research has shown that the storage and manipulation of cognitive symbol systems can be implemented as mathematical operations on fixed-length, high-dimensional vectors. These approaches are known as Vector Symbolic Architectures (VSAs; Gayler, 2003) and include Holographic Reduced Representation (HRR; Plate, 2003), MAP Coding (Gayler & Wales, 2000), Binary Splatter Codes (Kanerva, 1997), and others. Each of these provides an alternate method for converting symbols and symbol trees into vectors, combining vectors to perform symbolic manipulations, and extracting out the original components of that symbol tree.

In previous research we have shown how VSAs can be implemented in biologically realistic spiking neurons (Eliasmith, 2005; Stewart & Eliasmith, 2008). This approach is many orders of magnitude more efficient[1] than alternate theories of how symbolic manipulations could be performed by the brain (Stewart & Eliasmith, in press). However, one common criticism is that this approach does not yet show how these systems can clean up their representations. Performing symbol manipulations using VSAs is an inherently noisy process, and these operations must be performed by spiking neurons, adding a significant amount of random variation. When symbols are extracted from a bound representation, the brain needs a reliable method for identifying which symbol it is, allowing it to respond appropriately.

The purpose of this paper is to present an autoassociative memory constructed from spiking neurons which is appropriate for cleaning up the representations resulting from cognitive manipulations using VSAs. We first describe the characteristics of VSAs that define the statistical properties of the noise that must be removed. Next, a general method is described for encoding (and decoding) high-dimensional vectors across a population of spiking neurons. We then show that standard approaches to deriving connection weights have difficulty when scaled up to the number of symbols required for human. Our cleanup memory model is then presented, followed by an analysis of its behaviour.

Vector Symbolic Architectures

There are three core ideas for all VSAs. First, each symbol is represented by a randomly chosen vector. Second, two vectors can be combined by superposition (+) to produce a new vector that is *similar to* both of the original vectors. Third, two vectors can be combined by binding (⊗) to produce a new vector that is *dissimilar* to both of the original vectors, and this operation can be reversed by binding with the inverse of a vector (denoted with an underline), such that $\mathbf{A}\otimes\mathbf{B}\otimes\underline{\mathbf{B}}\approx\mathbf{A}$. These operations are similar to standard addition and multiplication in terms of being associative, commutative, and distributive. With such a system we can represent a structure such as `chase(dog,cat)` by performing the following calculation:

[1] For realistic vocabulary sizes, this approach uses three orders of magnitude fewer neurons than the Neural Blackboard Architecture (van der Velde & de Kamps, 2006) and seven orders of magnitude fewer than LISA (Hummel & Holyoak, 2003).

chase⊗verb + dog⊗subj + cat⊗obj

The result is a single vector of the same length as the vectors for the basic symbols (**chase**, **verb**, **dog**, etc.). This one vector can be interpreted as a representation of the entire structure because it is possible to extract the original components. For example, to determine the object of the above structure, we take the whole vector and bind it with the inverse of **obj**.

(chase⊗verb + dog⊗subj + cat⊗obj)⊗obj
= chase⊗verb⊗obj + dog⊗subj⊗obj + cat⊗obj⊗obj
≈ cat + chase⊗verb⊗obj + dog⊗subj⊗obj

The result is a vector that is similar to **cat**, but is not exactly the same since it has two additional terms superposed on it. Due to the properties of the binding operation, however, these two terms **chase⊗verb⊗obj** and **dog⊗subj⊗obj** will be vectors unlike any of the original symbols. They can thus be treated as randomly distributed noise. It is this noise that must be removed by the cleanup memory system.

While the above discussion applies to all VSAs, if we choose one particular type of VSA we identify the properties of the symbol and noise vectors. For this, we use Holographic Reduced Representations (HRRs; Plate, 2003). Here, each basic symbol vector is set by randomly selecting a point on the high-dimensional unit sphere (i.e. a random vector normalized to a length of one). Superposition is performed by vector addition and the binding operation is circular convolution.

The cleanup memory thus needs the following properties:

1) Recognize *M* unit vectors (one per symbol), distributed uniformly over a high-dimensional unit sphere.

2) Handle additive noise produced by adding *k* unit vectors uniformly distributed over the same sphere.

To be useful for cognitive operations, on the order of 100,000 symbols (*M*) must be able to be identified. The complexity of the structures that can be encoded is determined by *k*, indicating the number of terms that can be superposed and still lead to accurate recognition. This should be at least 7 ± 2 to conform to the standard chunk sizes used in cognitive modelling.

To determine whether recognition is accurate, we take the dot product of the correct vector and the output of the memory; if this value is above a threshold the symbol is successfully recognized. For the purposes of this paper, we arbitrarily choose a threshold of 0.7, although Plate (2003, p. 100) provides a method for determining the optimal threshold in special cases where *k* is fixed and known.

The final factor to consider when using Vector Symbolic Architectures is the number of dimensions. In an ideal case (where vectors are represented exactly, rather then via noisy spiking neurons), Plate (2003) derived the following formula for determining the minimum number of dimensions *D* required to represent combinations of k vectors out of *M* symbols and have a probability of error *q*:

$$D = 4.5(k+0.7)\ln(M/30q^4) \quad (1)$$

From this, we note that 700 dimensions would be sufficient to represent chunks of up to 7 symbols out of a vocabulary of 100,000 with an accuracy of 95%. However, this formula assumes an ideal cleanup memory.

Distributed Representation

There are a variety of methods whereby a numerical vector can be represented by a population of spiking neurons. The most simplistic approach is to have one neuron per dimension, and the firing rate of that neuron indicates the value in that dimension. However, this approach is highly fragile to neuron death and does not correspond to known methods of spatial representation by neurons. It is well established (e.g. Georgopoulos et al., 1986) that movement directions are encoded by having a large population of neurons, each of which is sensitive to a different direction. The firing rate of each neuron is related to the angle between that neuron's preferred direction vector and the value being encoded.[2]

We take this same approach to encode high-dimensional vectors. Each neuron has a preferred direction vector $\tilde{\phi}$ and the current entering the neuron is proportional to the dot product between this and the vector $\boldsymbol{x}$ being represented. If α is the neuron gain and J^{bias} is a fixed background current, then the total current flowing into cell *i* is:

$$J_i = \alpha_i \tilde{\phi}_i \cdot \boldsymbol{x} + J_i^{bias} \quad (2)$$

This current can be used as the input for any model of spiking neurons, such as the standard leaky integrate-and-fire (LIF) model. In general, $\boldsymbol{x}$ can vary over time as $\boldsymbol{x}(t)$ and the spikes produced will be based on this varying current. If the details of the neural model (i.e. the relation between input current and spiking behaviour) are written as $G[\cdot]$ and the neural noise of variance σ^2 is $\eta(\sigma)$, then the encoding of any given $\boldsymbol{x}(t)$ as the temporal spike pattern across the neural group is given as:

$$\sum_n \delta(t - t_{in}) = G_i[\alpha_i \tilde{\phi} \cdot \boldsymbol{x}(t) + J_i^{bias} + \eta_i(\sigma)] \quad (3)$$

Since this spiking pattern is meant to represent the original vector $\boldsymbol{x}$, it should be possible to determine an estimate $\hat{\boldsymbol{x}}(t)$ given only this spiking pattern. This can be done by deriving linearly optimal (in terms of minimizing squared error) decoding vectors ϕ for each neuron as per Equation 4, where a_i is the average firing rate for neuron *i* (see

[2] It should be noted that the simplistic representation mentioned initially is a special case of this approach, where the preferred direction vectors are exactly aligned along the dimensions being represented, rather than being randomly distributed.

Eliasmith & Anderson, 2003 for details). This method has been shown to uniquely combine accuracy and neurobiological plausibility (e.g. Salinas and Abbot, 1994).

$$\phi = \Gamma^{-1}\Upsilon \quad \Gamma_{ij} = \int a_i a_j \, dx \quad \Upsilon_j = \int a_i x \, dx \qquad (4)$$

To derive an estimate of $\boldsymbol{x}(t)$, we weight the decoding vectors by the post-synaptic current $h(t)$ induced by each spike. The shape and time-constant of this current are determined from the physiological properties of the neural group. The result is the best possible linear estimate of $\boldsymbol{x}(t)$ using only the spike timing information.

$$\hat{\boldsymbol{x}}(t) = \sum_{in} \delta(t - t_{in}) * h_i(t)\phi_i = \sum_{in} h(t - t_{in})\phi_i \qquad (5)$$

The representational error between $\boldsymbol{x}(t)$ and $\hat{\boldsymbol{x}}(t)$ is dependent on the particular neural parameters and encoding vectors, but in general is inversely proportional to the number of neurons in the group.

While the decoding vectors ϕ are useful for determining what a spike pattern represents, a more important feature is that they can also be used to derive optimal connection weights between neural groups. That is, consider a situation where one neural population represents $\boldsymbol{x}$ and we want a second neural population to represent $\boldsymbol{Wx}$ (where $\boldsymbol{W}$ is an arbitrary linear transformation). The optimal connection weights ω_{ij} between each neuron to achieve this are determined by Equation 6 (see Eliasmith & Anderson, 2003 for further details).

$$\omega_{ij} = \alpha_j \tilde{\phi}_j \boldsymbol{W} \phi_i \qquad (6)$$

These results provide a generic framework for representing vectors of any dimension using spiking neurons. These neurons can be made as realistic as possible (given computational processing constraints), including effects of adaptation, neurotransmitter re-uptake rates, refractory periods, and so on. Furthermore, we can derive the synaptic connection weights that will cause the neurons to perform the desired transformations on these represented values.

Standard Approaches

Given the above representation system, we have two groups of neurons: one representing the input (noisy) vector, and one representing the output (cleaned) vector. The goal then is to determine how to connect these neurons so as to achieve the best cleanup.

For this work, we are only considering feed-forward networks. That is, we do not consider models where activity flows backwards from the output to the input, or where the output is the same group of neurons as the input, but at a later time. These models, such as the Hopfield network, must wait for their output to "settle", requiring significantly more time than purely feed-forward models.

Linear Autoassociation

The simplest autoassociation memory merely performs a linear transformation on the input to produce the output (Hinton & Anderson, 1989). If the matrix $\boldsymbol{X}$ consists of a set of noisy vectors and the matrix $\boldsymbol{Y}$ holds the corresponding cleaned vectors, then we want to find $\boldsymbol{W}$ such that $\boldsymbol{WX} \approx \boldsymbol{Y}$. Given the subsequent noisy vector $\boldsymbol{x}$, it can then be multiplied by $\boldsymbol{W}$ to produce the estimated cleaned up item $\boldsymbol{y}=\boldsymbol{Wx}$. Once $\boldsymbol{W}$ is found, we derive the connection weights for this linear transformation using Equation 6.

A variety of methods exist to find the $\boldsymbol{W}$ that minimizes the error between $\boldsymbol{WX}$ and $\boldsymbol{Y}$. Figure 1 shows the result of using the Penrose-Moore pseudoinverse, which was chosen since $\boldsymbol{X}$ is generally not full rank.

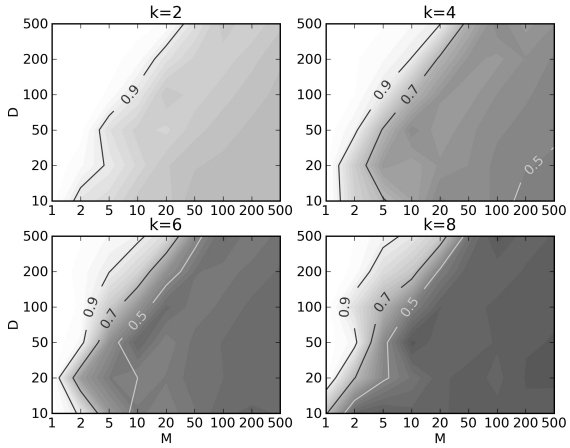

Figure 1: Accuracy of the linear autoassociation network for varying *D*, *M*, and *k*. Values above 0.7 (shown in lightest shading) indicate successful cleanup (i.e. output values sufficiently close to the original non-noisy vector).

These results show that the linear association approach does not scale up for large values of *M*. In 500 dimensions this network is unable to accurately clean up a vector where 4 symbols are combined if there are more than 50 possible symbols. This is much smaller than the desired 100,000.

Linear Neural Transformation

A second possibility is to directly determine the optimal connection weights, rather than relying on Equation 6. Here, instead of $\boldsymbol{X}$ being the noisy vectors, it is the spiking rate of the individual neurons when representing those vectors. This approach is used extensively in the Neural Engineering Framework (Eliasmith and Anderson, 2003) to derive synaptic connection weights that can perform nonlinear operations, using a slight modification of Equation 4 where $\boldsymbol{x}$ is replaced by the corresponding cleaned up vector. This allows synaptic connection weights to be derived that approximate arbitrary nonlinear functions.

While the results in Figure 2 show that this approach is a significant improvement over Figure 1 in terms of handling larger values of *k* at smaller *D*, it is still not scaling up for larger values of *M*.

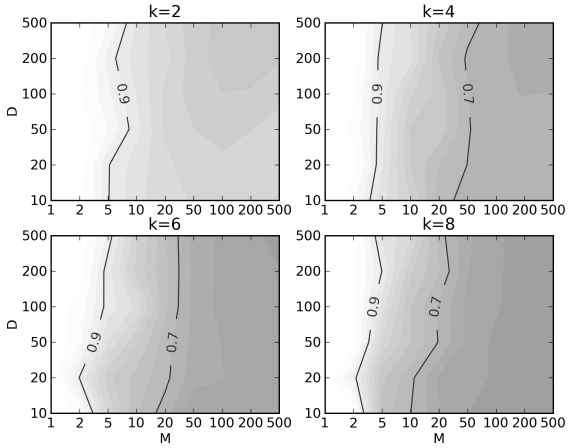

Figure 2: Accuracy of the linear autoassociation approach applied to individual neuron firing rates for varying *D*, *M*, and *k*. Values above 0.7 indicate successful cleanup.

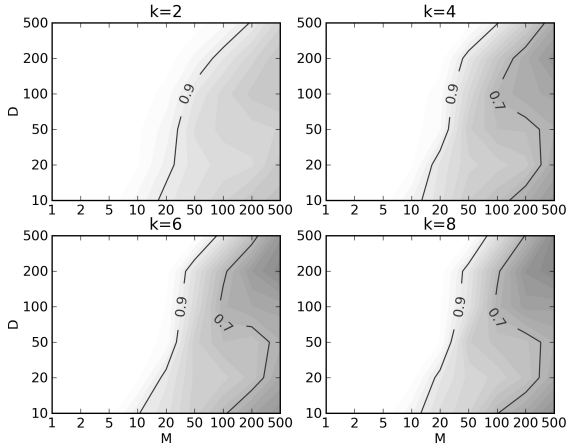

Figure 3: Accuracy of the multilayer perceptron for varying *D*, *M* and *k*. Values above 0.7 indicate successful cleanup.

Multilayer Perceptron

One potential reason for the failure of the linear associator discussed in the previous section is that the function being computed is highly nonlinear. To address this, we can make use of a multilayer perceptron, capable of computing much more complex functions. This involves introducing a new hidden layer of neurons between the input and output.

The multilayer perceptron is the most famous and widely used artificial neural network (Rumelhart et al., 1986). Using a two layer MLP, a mapping is learned to convert noisy input vectors into their cleaned (or prototype) vectors.

Instead of directly calculating the weights for these networks, a learning rule (such as the classic backpropagation of error rule) must be used. This allows the system to find a suitable intermediate representation in the hidden layer which makes the cleanup operation most accurate. For this task we trained the MLP using gradient descent on the sum of the squared error.

In theory, given enough time, hidden nodes, and a sufficiently powerful optimization algorithm, this approach should be able to find the optimal synaptic connection weights to perform this task. However, as the results in Figure 3 show, due to limited computational resources we were unable to successfully train this network for large M. This is in part due to the fact that the MLP requires many more hidden nodes than the vector dimension in order to generalize across the entire input domain.

More importantly, the standard strengths of a backpropagation network are not applicable to the cleanup task. Crucially, there is no inner structure in the data being modelled; each symbol is a randomly chosen unit vector. This means that the network cannot use its hidden layer to form an internal representation that simplifies the task.

Overall, it is likely possible to improve on this approach to training a network to perform cleanup. However, such a method may require significantly larger amounts of computing resources as *M* increases.

A Cleanup Memory Model

From the MLP model, it is clear that while transforming the initial representation through a middle layer of neurons can provide a significant improvement, it is impractical to learn the required synaptic connection weights. Instead, for our cleanup memory model we choose to directly derive the optimal weights. To do this, we first identify how we want the middle layer of neurons to respond. This involves defining their preferred direction vectors $\tilde{\phi}$, gain α, and J^{bias} as per Equation 2. Given these, we can use Equation 6 to derive the neural connection weights that will result in this behaviour. Since no transformation of the vector itself is to be performed by the weights, $\boldsymbol{W}$ in Equation 6 is set to be the identity matrix.

For the preferred direction vectors, we choose exactly those vectors that must be cleaned up. For redundancy, we have ~10 neurons for each of the *M* vectors, meaning that there are particular neurons that fire maximally for each symbol. Furthermore, we set J^{bias} to be slightly negative for each neuron. The resulting connection weights ω_{ij} cause the middle layer neurons to only fire if the dot product of the input vector with the corresponding clean vector is greater than some small threshold (0.2).

In effect, the inherent non-linearity of the neurons (the fact that they do not fire if their input current is too low) is being used to perform cleanup. This middle layer is good at representing the cleaned vectors, but is poor at representing small vectors in any of those directions. Since the noise added to the input consists of randomly chosen vectors, these will generally have small dot products with each of the preferred direction vectors, and so will not cause sufficient activation for the neuron to fire. The presence of a slight background inhibition (the negative J^{bias}) allows the neurons to be insensitive to the noise.

The firing rates of ten sample middle layer neurons are shown in Figure 4. Their activity varies as the dot product of the input and the neurons' preferred direction vector changes.

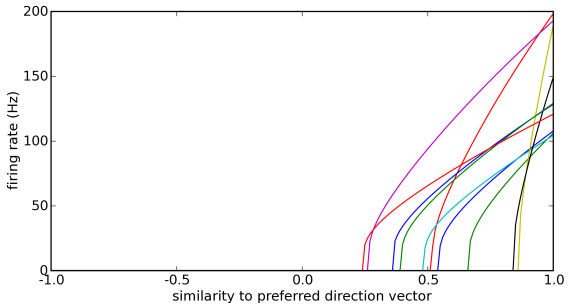

Figure 4: Middle layer neuron tuning curves. Average firing rates for ten neurons are shown as the input to the cleanup memory changes. Similarity is the dot product of the input vector with the preferred direction vector.

Given this middle layer representation we can then calculate the optimal connection weights with the output neural group. This output group can have any arbitrarily chosen preferred direction vectors $\tilde{\phi}$ and other neural properties. Equation 6 is used to calculate these weights, again setting $\boldsymbol{W}$ to be the identity matrix.

Performance

We evaluated this implementation of cleanup memory in the same manner as the previous models and the results are shown in Figure 5. It should be noted that these graphs extend to much larger M (10,000 symbols rather than 500) than the previous figures.

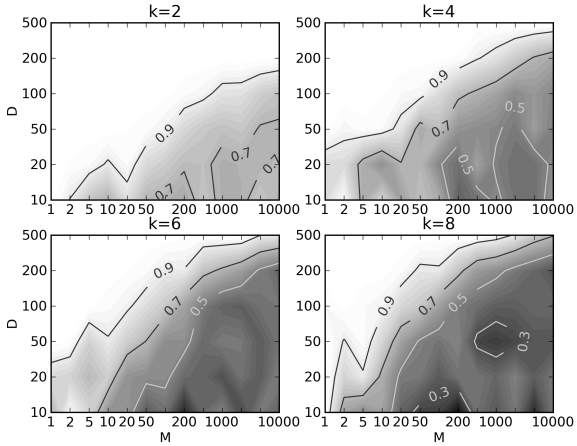

Figure 5: Accuracy of our neural cleanup memory for varying D, M and k. Values above 0.7 indicate successful cleanup.

Importantly, our neural cleanup memory system was able to successfully cleanup combinations of 8 symbols out of a vocabulary of 10,000 using 500 dimensional vectors. Furthermore, its capabilities increase rapidly with the number of dimensions. We have evaluated this model up to M=100,000 and D=1000, producing consistently high quality cleanup results.

We have thus demonstrated an effective implementation of a neural autoassociator as a cleanup memory for Vector Symbolic Architectures. The number of neurons required for cleanup scales linearly with M, while the number of neurons required for storing the resulting cleaned vector is linear in D.

Comparison to the Ideal

To determine how closely our model approaches ideal behaviour (even though it is implemented using realistically noisy spiking neurons), we can examine the recognition behaviour of a perfect mathematical cleanup system. This is used by Plate (2003) in his analysis of the Holographic Reduced Representation form of VSA, and merely outputs the clean vector that is closest to the input noisy vector. This ideal system can be approximated by Equation 1, and its actual behaviour is shown in Figure 6.

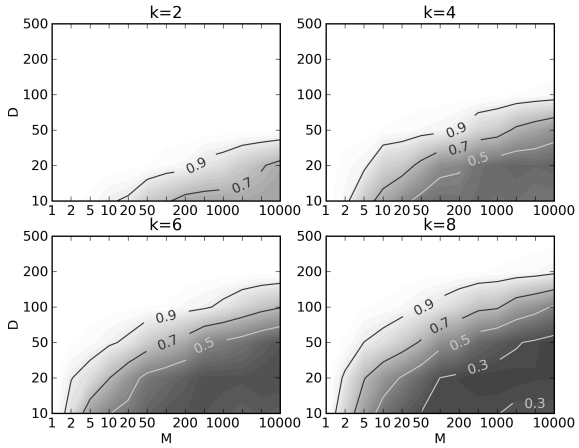

Figure 6: Accuracy of an ideal cleanup memory for varying D, M and k. Values above 0.7 indicate successful cleanup.

From this result, we see that our neural cleanup memory and the ideal cleanup both exhibit a similar growth in representational capacity as the dimensionality of the vectors increases. While the neural version is less accurate, it still is able to scale up to large M. This ability is not seen in the cleanup models examined previously.

Dynamics and Timing

Since a cleanup memory is meant to be a component to support symbolic manipulations by spiking neurons, it must not only be efficient in terms of numbers of neurons, but also in terms of the amount of time required to perform clean up. This is why we did not consider models that require a long settling time (such as a Hopfield network).

Since the dynamics of the neurons in our model (G in Equation 3) can be adjusted to match those of real neurons, we can generate predictions as to how the output of the cleanup memory will vary over time. Even with a constant input vector $\boldsymbol{x}$, the actual value being represented by the output of the cleanup memory will vary since it is decoded from the spike train as per Equation 5.

The precise timing characteristics of the neural model will vary based on the neural parameters. We used typical values for cortical neurons: a refractory period of 2ms, a membrane time constant of 20ms, and a maximum firing rate of 200Hz. We applied random noise in the input current to each cell of σ=10% (see Equation 3). We also assumed NMDA neurotransmitter receptors, giving a time constant of 5ms for the post-synaptic current (*h(t)* in Equation 5).

To observe the dynamics, we ran a cleanup memory with D=500, M=10,000, and k=8. Over the course of 250ms of simulated time, we input five different noisy vectors for 50ms each. The output from the system was measured at each time step. Figure 7 shows the result of comparing the output of the model (the cleaned up vector) with the corresponding five original vectors. As in the rest of this paper, comparison was done by the dot product of the output vector and the desired clean vector.

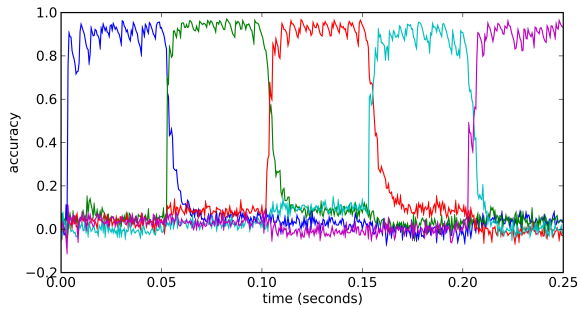

Figure 7: Temporal accuracy of the cleanup memory. Five noisy vectors are presented for 50msec each. Graphed lines show the dot product of the output of the network and the five original clean vectors.

These results indicate that the network reliably cleans the input vector and does so within 5-10 milliseconds. This makes our cleanup memory suitable for fast recognition, which is needed for symbolic manipulations at a cognitive time scale.

Conclusions

The model presented here is the first demonstration that a cleanup memory can be efficiently implemented by realistic spiking neurons. The number of neurons required to build this memory increases linearly in the number of distinct symbols that can be recognized. The accuracy approaches that of an ideal mathematical cleanup, and can perform cleanup in 5-10ms using realistically noisy spiking neurons.

Previous research (e.g. Eliasmith, 2005) has demonstrated realistic neurons performing the binding and superposition operations required for Vector Symbolic Architectures. Given the cleanup memory presented here, arbitrary symbol structures can be encoded, transformed, and recognized, all within a spiking network. As a result, we take this work to complete the currently most biologically plausible implementation of a symbolic cognitive architecture (Stewart & Eliasmith, 2009).

References

Anderson, J. R. and Lebiere, C. (1998). *The Atomic Components of Thought*. Mahwah, NJ: Erlbaum.

Eliasmith, C. (2005). Cognition with neurons: A large-scale, biologically realistic model of the Wason task. *Proceedings of the 27th Annual Meeting of the Cognitive Science Society*.

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

Gayler, R. W., & Wales, R. (2000). Multiplicative binding, representation operators and analogical inference. *5th Australasian Cognitive Science Conference.*

Gayler, R. (2003). Vector symbolic architectures answer Jackendoff's challenges for cognitive neuroscience. *ICCS/ASCS International Conference on Cognitive Science*.

Georgopoulos, A.P., Schwartz, A.B., & Kettner, R.E. (1986). Neuronal population coding of movement direction, *Science 233*(4771), 1416-1419.

Hinton, G., & Andersen, J. (1989). *Parallel Models of Associative Memory*. Lawrence Erlbaum Associates, Inc.

Hummel, J. E., & Holyoak, K. J. (2003). A symbolic-connectionist theory of relational inference and generalization. *Psychological Review*, 110(2), 220-264.

Rumelhart, D., Hinton, G., and Williams, R. (1986). Learning representations by back-propagating errors. *Nature 323*, 533-536.

Kanerva, P. (1997). Fully distributed representation. *Proceedings of 1997 Real World Computing Symposium.*

Plate, T. (2003). *Holographic reduced representations.* Stanford, CA: CSLI Publication.

Pollack, J. B. (1988). Recursive auto-associative memory: devising compositional distributed representations. *Proceedings of the 10th Annual Conference of the Cognitive Science Society.*

Salinas, E., Abbott, L.F. (1994). Vector reconstruction from firing rates. *Journal of Computational Neuroscience 1*, 89-107.

Stewart, T.C. and Eliasmith, C. (2008) Building Production Systems with Realistic Spiking Neurons. *Proceedings of the 30th Annual Meeting of the Cognitive Science Society.*

Stewart, T.C. and Eliasmith, C. (2009). Compositionality and Biologically Plausible Models. In W. Hinzen, E. Machery, and M. Werning (Eds.), *Oxford Handbook of Compositionality*. Oxford University Press.

Sun, R. (2006). The CLARION cognitive architecture: Extending cognitive modeling to social simulation. In Ron Sun (ed.), *Cognition and Multi-Agent Interaction.* New York: Cambridge University Press.

van der Velde, F., & de Kamps, M. (2006). Neural blackboard architectures of combinatorial structures in cognition. *Behavioral and Brain Sciences*, 29, 37-70.