# Parsing Sequentially Presented Commands in a Large-Scale Biologically Realistic Brain Model

**Terrence C. Stewart (tcstewar@uwaterloo.ca)**
**Chris Eliasmith (celiasmith@uwaterloo.ca)**
Centre for Theoretical Neuroscience, University of Waterloo
200 University Avenue West, Waterloo, Ontario N2L 3G1 Canada

**Abstract**

We present a neural mechanism for interpreting and executing visually presented commands. These are simple verb-noun commands (such as WRITE THREE) and can also include conditionals ([if] SEE SEVEN, [then] WRITE THREE). We apply this to a simplified version of our large-scale functional brain model "Spaun", where input is a 28x28 pixel visual stimulus, with a different pattern for each word. Output controls a simulated arm, giving hand-written answers. Cortical areas for categorizing, storing, and interpreting information are controlled by the basal ganglia (action selection) and thalamus (routing). The final model has ~100,000 LIF spiking neurons. We show that the model is extremely robust to neural damage (40% of neurons can be destroyed before performance drops significantly). Performance also drops for visual display times less than 250ms. Importantly, the system also scales to large vocabularies (~100,000 nouns and verbs) without requiring an exponentially large number of neurons.

**Keywords:** neural engineering; parsing; cognitive control; spiking neurons; whole-brain systems; cognitive architecture

## Large-Scale Functional Brain Modelling

Our goal is to produce models of human cognition that are specified down to the neural level. That is, we want to know how the low-level neural details (including spikes and various neurotransmitters) give rise to human behaviour via their complex interconnections and interactions. We have previously published our first step in this direction, which is currently the world's largest functional brain model (Eliasmith et al., 2012). This model, "Spaun", has 2.5-million spiking neurons, includes twenty different brain areas, and can perform eight different cognitive tasks (including digit recognition, list memory, addition, and pattern completion). Input is through a single eye with a 28 by 28 retina, and the output controls a simulated three-joint six-muscle arm, allowing it to write its answers. Spaun is told what task to perform via its visual input, so it must selectively re-route information between brain areas as appropriate for different tasks. This uses the cortex-basal ganglia-thalamus loop, where the basal ganglia performs action selection by comparing the current brain state to the ideal brain state for each action, and the chosen action activates cortical communication channels via the thalamus.

One limitation with Spaun is that it cannot learn new tasks. The eight tasks it can perform are set by the synaptic connections between cortex and basal ganglia. To address this, the work presented here adds a new general-purpose task for Spaun: one where it can be visually presented with commands for it to follow.

## Parsing Visual Commands

To provide new instructions to a model that only has a visual input, we need the model to process a sequential set of images and convert that into an internal representation of a command. This is a simplified language comprehension task, within a fairly restricted domain.

Basic commands can be thought of as verb-noun pairs, such as WRITE NINE. However, because the visual system is limited to 28x28 pixels, it does not have the visual acuity to interpret full words at a time. Rather than flashing each letter in each word up individually (a fairly non-typical reading strategy), we use a single symbol for each word, and present those symbols sequentially. So, for the command WRITE NINE, we present a "W" followed by a "9".

Valid commands are limited by the set of basic actions that the model knows how to do. While the full Spaun model can perform many operations including mental arithmetic, keeping track of elements in a list, and pattern completion, for simplicity in this paper we only consider the actions WRITE (W), REMEMBER (R), and INCREASE (C). For example, the model could be told to remember a two, increase it, and write the result ("R 2, C #, W #", where # is a general-purpose indexical referring to the number currently being remembered, and a comma is a slight pause between instructions). The correct result from this command would be the written number 3.

Furthermore, instructions can also include conditional clauses based on what the model can currently SEE ("S"). For example, "S 4, W 9" is interpreted as "if you see a four, then write a nine". To do this, the model must be capable of representing structured relationships.

The goal of this work is to give a spiking neuron implementation of this parsing process, integrated within an existing spiking neuron model of the rest of the brain (including vision, motor, working memory, and cognitive control areas). To argue that this is a plausible model, we show that a) its performance degrades gracefully as neurons die, b) it fails on uninterpretable grammatical structures, and c) it scales to human-sized vocabularies, dealing with the exponential growth of vocabulary combinations.

That said, there are considerable limitations to this work. It does not deal with token separation, since the symbols are shown to it one at a time. It also does not handle ambiguous terms (all symbols have exactly one meaning). We are also not specifying the full developmental and learning process that results in this model (although existing learning rules could be used, given a detailed error signal and large amounts of time).

## The Neural Engineering Framework

The Neural Engineering Framework (NEF; Eliasmith & Anderson, 2003) transforms a high-level description of the *variables* being represented and the *operations* on those variables into a detailed spiking-neuron model subject to neurobiological constraints.

In the NEF, neurons are organized into groups, and each group forms a distributed representation of a particular variable. Different patterns of activity across the group correspond to different values for that variable. These values are, in general, vectors, so a particular group of 2,000 neurons might represent a 64-dimensional variable. While the NEF supports any neuron type, for this paper we use standard leaky integrate-and-fire (LIF) neurons whose parameters (refractory period, capacitance, neurotransmitter time constant, etc.) are set to be consistent with the details of the particular brain regions being modelled.

Within a population of neurons, each neuron has a particular *preferred direction vector*. This is the vector for which that neuron will fire most strongly. These vectors $\boldsymbol{e}$ are randomly chosen along with the neuron gain $\alpha$ and bias $J_{bias}$ to produce a heterogeneous population of neurons. The current flowing into a neuron when representing a vector $\boldsymbol{x}$ is given by Equation 1.

Given a pattern of activity, we can estimate the currently represented $\boldsymbol{x}$ value as $\sum d_i a_i$ where $a_i$ is the neuron activity and $\boldsymbol{d}$ is a *decoder* given by Equation 2. This is the optimal least-squares linear estimate of $\boldsymbol{x}$ (the value being represented) given $a$ (the current activity of the neurons).

The key part of the NEF is that this decoder also allows us to determine the synaptic connection weights between neural groups that will force them to compute a desired function. For example, if we want to connect a neural group representing $\boldsymbol{x}$ to a neural group representing $\boldsymbol{y}$ such that $\boldsymbol{y}=\boldsymbol{Mx}$ (where $\boldsymbol{M}$ is an arbitrary matrix), then the synaptic connection weights between neuron $i$ in the first population and neuron $j$ in the second are given by Equation 3.

For connections that compute nonlinear functions, we adjust Equation 2 slightly as given in Equation 4. This finds a decoder that approximates the arbitrary function *f(x)*.

$$J=\alpha e\cdot x+J_{bias} \tag{1}$$

$$d=\Gamma^{-1}Y \quad \Gamma_{ij}=\int a_i a_j dx \quad Y_j=\int a_j x dx \tag{2}$$

$$\omega_{ij}=\alpha_j e_j M d_i \tag{3}$$

$$d^{f(x)}=\Gamma^{-1}Y \quad \Gamma_{ij}=\int a_i a_j dx \quad Y_j=\int a_j f(x) dx \tag{4}$$

It should be noted that the *accuracy* with which neurons will perform the desired computation using this technique is dependent on many factors. This includes the neuron properties such as overall firing rate and their membrane time constant. Accuracy can be increased arbitrarily by increasing the number of neurons (but, of course, to be realistic we are constrained by the number of neurons in the brain). In general, discontinuous functions are very difficult for neurons to approximate.

## Symbol-Like Processing with Spiking Neurons

While the NEF allows us to convert algorithms that use vectors and functions into spiking neuron models, a further technique is needed to handle the symbol manipulation that is the hallmark of cognitive activity. This is especially important for parsing and representing complex commands.

The core idea is to have a particular vector for each atomic symbol that can be represented. For this paper, these vectors are chosen randomly, but they can also be chosen such that semantically related symbols have similar vectors. To combine symbols, we perform computations on these vectors, giving new vectors that represent the combination.

There are a variety of computations that can be used to combine these vectors (Gayler, 2003), but for our model we follow Plate (2003). Here, symbols can be combined by vector addition (**+**) and circular convolution (⊛). Both operations are accurately approximated by the NEF method.

To demonstrate how this system works, consider representing the command "If you see a 9, write an 8". A simplistic approach would be to take vectors for all the atomic concepts (**SEE**, **NINE**, **WRITE**, and **EIGHT**) and add them together to represent the full sentence (**SEE+NINE+WRITE+EIGHT**). However, this does not work, since the resulting sentence loses all order information. In particular, the command "If you see an 8, write a 9" would result in *exactly the same vector*.

To deal with this, we use circular convolution (⊛) and introduce new vectors for denoting structural information. The ⊛ operator takes two vectors and produces a new vector that is highly dissimilar to the original two. So instead of **WRITE+EIGHT** we can do **VERB⊛WRITE+ NOUN⊛EIGHT**. Furthermore, we can nest this process to make more complex phrases. The full command can be represented by the vector **S=CONDITION⊛(SEE⊛NINE)+ VERB⊛WRITE+NOUN⊛EIGHT**.

Importantly, given this vector **S** that represents a full command, we can extract out the individual components. Plate (2003) showed that a simple re-arranging of the elements of a vector makes an approximate inverse operation. For example, if we want to know the main verb in **S**, we compute **S⊛*inv*(VERB)**. The result will be approximately **WRITE**. The accuracy of this approximation depends on the number of terms being added and the dimensionality of the vectors. In particular, as the number of dimensions increases, there is an exponential growth in representational capacity.

We refer to these vectors as *semantic pointers*. They are semantic in that the vector itself has meaning about the whole. Most usefully, the similarity between vectors indicates the similarity of the full structure. **WRITE** commands will have a higher degree of similarity to each other than to other commands. Furthermore, they are pointers in the computer science sense because they can be dereferenced, recovering (an approximation of) the original data. Semantic pointers are compressed representations, in the same way that vision models can be thought of as compressing an image into a high-level representation.

## Vision

For vision, we adapt a Deep Belief Network (Hinton, 2010). The input is a 28 by 28 pixel retinal image, which is then processed by four different cortical layers. Each layer learns to extract and compress the regular patterns in the layer before it. We convert this model to spiking neurons by simulating each neuron in the DBN with ten realistic spiking neurons and using Equations 3 and 4 to solve for the connection weights that approximate the original model.

The output from the DBN (inferior temporal cortex) must then be mapped to a semantic pointer. One way to perform this mapping would be to use an associative cleanup memory (Stewart, Tang, & Eliasmith, 2011), which scales to hundreds of thousands of items but requires additional neurons to recognize each item. For simplicity, here we use no additional neurons, but rather compute an approximate mapping between the compressed representation of the visual stimulus and the desired semantic pointers (Equation 5), where $v_i$ is the average output of the Deep Belief Network for a particular category (all the 3's, for example), and $s_i$ is the corresponding semantic pointer (**THREE**). As always, Equations 3 and 4 give the synaptic connections.

$$f(x) = x \cdot V \qquad V = \sum v_i \times s_i \tag{5}$$

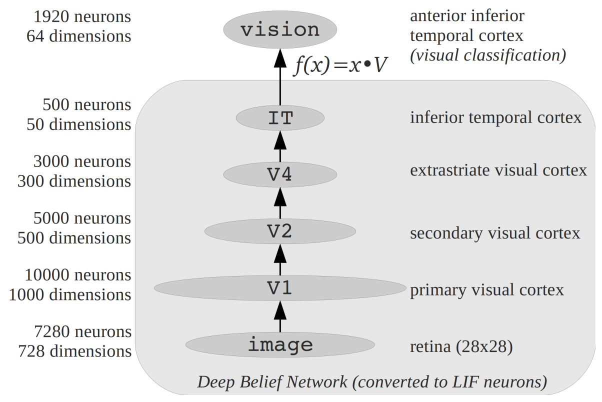

Figure 1: Converting visual input into the correct semantic pointer. For example, showing an image of a 7 to the retina will produce the vector **SEVEN** in the vision population.

## Simple Parsing

A first step towards parsing commands is to identify and store noun-verb pairs. That is, given a visual input of **WRITE** followed by **THREE**, we need one group of neurons to represent the verb (**WRITE**) and another to represent the noun (**THREE**). The outputs from these groups then drive a third group of neurons to represent the full phrase **VERB⊛WRITE+NOUN⊛THREE**. This is accomplished by connecting the `verb` population to the `phrase` population with connection weights optimized to perform the function *f*(**x**)=**x**⊛**VERB**. Similarly, the `noun` population's connection is optimized for the function *f*(**x**)=**x**⊛**NOUN**. As with all synaptic connections in this model, this optimization is performed using Equations 3 and 4.

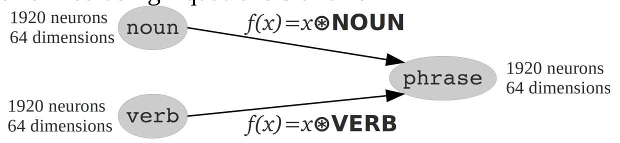

Figure 2: 5760 spiking neurons combining two arbitrary 64-dimensional vectors (`noun` and `verb`) into a single 64-dimensional vector `phrase`**=**`verb`**⊛VERB+**`noun`**⊛NOUN**.

In order for this system to work in the context of an overall brain model, we need a mechanism to selectively route information from visual areas to the two populations. When the system sees **WRITE** it should pass this vector into the `verb` neurons, and when it sees **THREE** it should pass that vector to the `noun` neurons. This sort of selective routing of information is exactly what the cortex-basal ganglia-thalamus loop is believed to perform. We use our existing model for this loop, which is based on our spiking version of a model of action selection in the basal ganglia (Stewart, Choo, & Eliasmith, 2010).

The basal ganglia selects between two actions. One action is to route information from the `vision` system to the `noun` population, and the other is to route that same information to the `verb` population. To perform these actions the output from the basal ganglia goes to the thalamus, where it releases the inhibition on the desired communication channel. (A communication channel is a connection that computes the function *f*(**x**)=**x**). If this is inhibited, the neurons do not fire, and so no information is passed. Selecting the action releases the inhibition, allowing the information to flow.

To decide which action to perform, the inputs to the basal ganglia must compute the utility of the two actions. For the first action, this is a function that outputs a 1 if there is a noun in `vision`, and a 0 otherwise. The second action is similar, but for identifying verbs. This is a simple classifier and can again be expressed as a function whose connection weights are computed with Equations 3 and 4.

The resulting system is capable of taking a stream of visual stimulus as input and keeping track of the most recent noun and the most recent verb seen. This verb and noun are then combined into a single vector representing that pair.

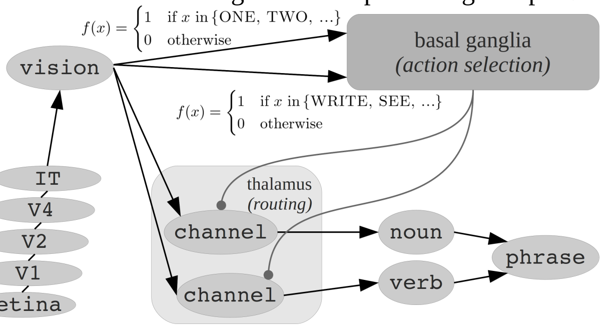
$f(x) = \begin{cases} 1 & \text{if } x \text{ in } \{\text{ONE, TWO, ...}\} \\ 0 & \text{otherwise} \end{cases}$
basal ganglia
(action selection)
vision
$f(x) = \begin{cases} 1 & \text{if } x \text{ in } \{\text{WRITE, SEE, ...}\} \\ 0 & \text{otherwise} \end{cases}$
IT
V4
V2
V1
retina
thalamus
(routing)
channel
channel
noun
verb
phrase

Figure 3: Routing information from vision to the correct `noun` and `verb` populations, depending on whether the visual stimulus is categorized as a noun or a verb. Once routed, the correct combined `phrase` is computed as in Figure 2.

## Responding to Commands

Once the model has formed a single representation of the command itself, we also need to show that it can execute that command correctly. That is, not only can the representation be encoded by spiking neurons, but it can also be decoded by spiking neurons to perform a task.

Since the focus of this paper is the parsing of a command, we use a very simple system for executing commands. In other work we show how to process significantly more complex commands, (Choo & Eliasmith, submitted), but those commands are directly injected into the brain model, rather than being presented visually and parsed.

In this case, performing a command occurs via a new action added to the basal ganglia. It has a high utility when there is no visual input and when the `phrase` is similar to **VERB⊛WRITE**. When selected, this action routes the information from `phrase` to `motor` while convolving it with the inverse of **NOUN**. Thus, if the phrase is **VERB⊛WRITE+NOUN⊛FOUR**, the semantic pointer **inv(NOUN)⊛(VERB⊛WRITE+NOUN⊛FOUR)** will be routed to the `motor` area. Since **NOUN** and **inv(NOUN)** approximately cancel, the value set to the `motor` area will be close to the ideal vector for **FOUR**. This is mapped to a series of hand positions using the same method as Eq. 5.

The behaviour of the model is shown in Figure 5. The visual input is shown in the top row, and the written responses are shown in the bottom row. The other rows show the spiking behaviour of 50 neurons from each of the key brain areas in the model.

The first thing to note is that the model performs accurately. The correct response is given for each case. Furthermore, it should be noted that the two words in the command can be given in either order (WRITE FIVE versus TWO WRITE). This is because we have not imposed a particular grammatical order. While it would be possible to do so, we note that English speakers are quite capable of

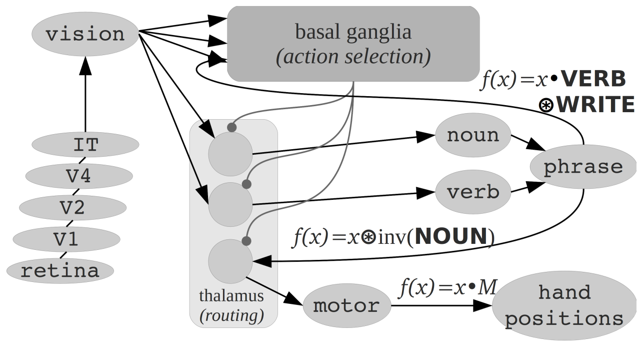

Figure 4: Executing a WRITE action

correctly interpreting TWO WRITE as a command. However, as demonstrated in the section on Conditional Statements, word order does matter for complex commands.

The spike patterns shown in Figure 5 provide some insight into the performance of the model. In the `vision` row, we can see different patterns of activity for each visual input, as expected. The pattern for "W" and the pattern for seeing nothing at all are quite distinct. Similarly, the spike patterns in the `noun` and `verb` populations change depending on which term is currently being memorized, and these patterns in turn affect the `phrase` population.

Another feature that can be seen in Figure 5 is the cognitive reaction time. Each symbol is shown for 0.5 seconds, but the `motor` output is clearly delayed slightly. For example, the visual input is cleared at t=1.0s, but the spiking behaviour in `motor` doesn't change until t=1.05s. This is the time required for the model to notice the change in visual input, perform action selection in the basal ganglia, release the inhibition in the thalamus, and allow the information from the phrase neurons to pass to the motor neurons. The exact amount of time required is a function of the connectivity and neurotransmitter time constants, all of which are taken from neurological data (so they are not free parameters). For more analysis of this feature of the basal ganglia model, see (Stewart, Choo, & Eliasmith, 2010).

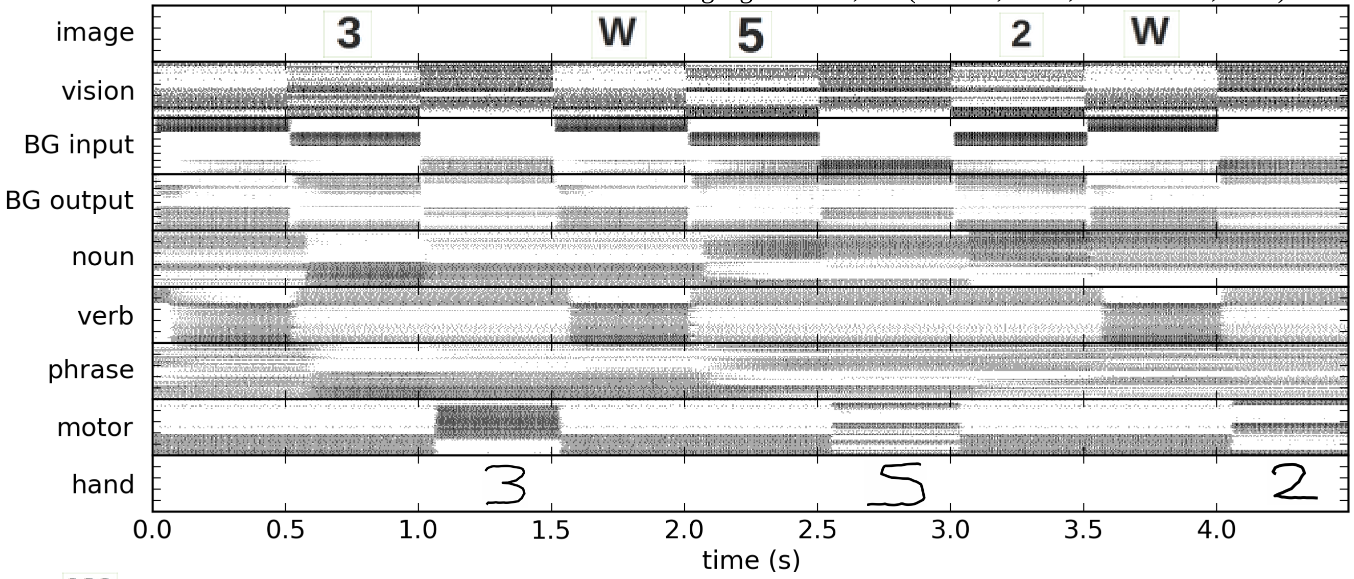

Figure 5: Behaviour of the model given the commands WRITE THREE, WRITE FIVE, and TWO WRITE

## Memory

To demonstrate that this system can parse commands other than WRITE <number>, we add a memory action. If the model is told to REMEMBER 4 (“R 4”), the `phrase` will be similar to **VERB⊛REMEMBER**. We add an action for this condition that routes the `phrase` information to the `memory` while transforming it by **inv(NOUN)**. As with the **WRITE** action, this extracts the **FOUR** from the `phrase`. For this action, the output vector is routed to a working memory area. This is a group of neurons that stores a vector (like every other group of neurons in the model), but that has a communication channel *back to itself*. This recurrent connection causes the neurons to maintain their own state after the input is removed. This structure has been shown to match neural behaviour of visual working memory (Singh and Eliasmith, 2006), and is stable over long periods of time (tens of seconds).

Finally, we show that we can extract information from working memory by adding a special write action WRITE NUMBER (“W #”). This action has a high utility when the phrase is **VERB⊛WRITE+NOUN⊛NUMBER** and causes the information in working `memory` to be routed to the `motor` system.

The result (Figure 6) is a system that can respond correctly to two different verbs and ten nouns (only **ZERO** through **NINE** were implemented). Importantly, adding the new action did not require any modifications to the `phrase` population. This is due to the fact that the semantic pointers used to represent the phrase are simply fixed-length vectors, and the `phrase` population is capable of storing any vector. No modifications to that population are needed to let it store a new vector like **VERB⊛REMEMBER+NOUN⊛THREE**, even if it has never seen it before.

In other words, the model does not require an exponential growth in numbers of neurons in order to handle the exponential growth in possible phrases that it can correctly interpret. Adding new actions only requires adding the neural populations needed to perform that action (in this case, the working memory population) and new connections between existing populations (in this case, `phrase`) and the basal ganglia and thalamus.

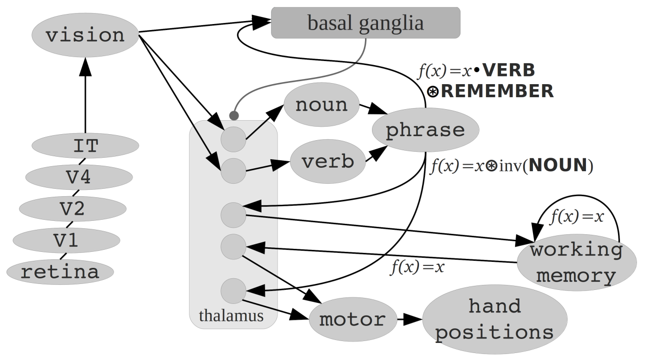

Figure 6: A model with **REMEMBER** and **WRITE** actions. Given an input REMEMBER SIX <long pause> WRITE NUMBER it will write the number 6.

## Model Performance

The behaviour of this model in a variety of conditions is shown in Figure 7. For each condition, new neurons (with preferred direction vectors, gains, and background currents) were generated, and Equations 3 and 4 were used to solve for all the synaptic connection weights.

First, the maximum vocabulary size (the largest number of nouns such that there is still a 95% chance of correctly responding) scales exponentially as the number of neurons per population increases. This is an expected consequence of vector representations (e.g., Plate, 2003), as the number of dimensions accurately represented scales linearly with the number of neurons, while the volume of a hypersphere scales exponentially with the number of dimensions.

Second, the model is robust to destruction of neurons. To simulate neural death, we randomly delete neurons from every population, and then re-use Equations 3 and 4 to compute new connection weights between the remaining neurons. Performance decreases, but remains above well above chance until less than 40% of the neurons remain. This shows a gradual degradation in behaviour, rather than catastrophic failure.

Finally, we show the model performs well for varying stimulus presentation times, but poor performance when symbols are seen for less than 250 milliseconds.

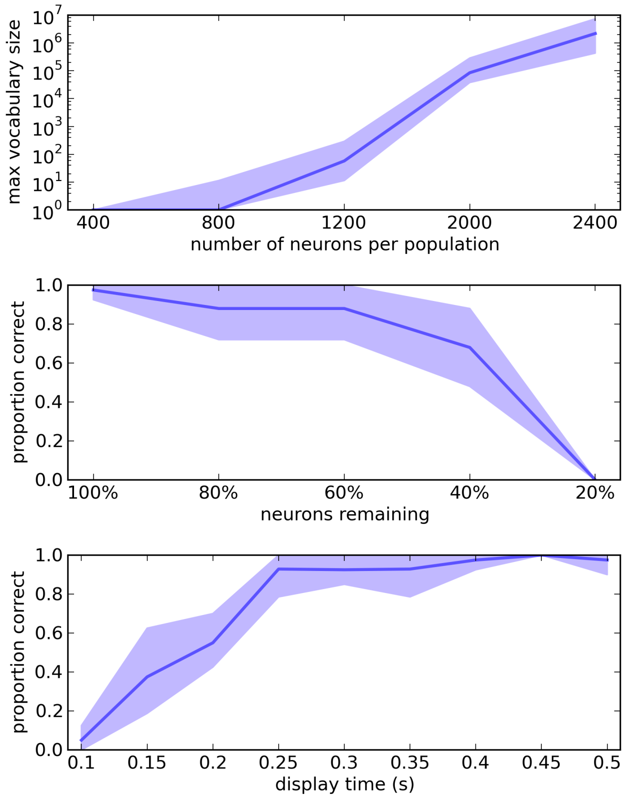

Figure 7: Model performance for varying vocabulary sizes, neural destruction, and display times. Input is of the form “R <number>, W #”, and an output is judged correct if the model writes the correct number. Shaded area is the 95% bootstrap confidence interval over 50 trials.

## Conditional Statements

The method used to add the **REMEMBER** action can be used to add many new actions. For instance, an **INCREMENT** action can be added which increases the number stored in `memory`. However, to show that this method extends to more complex rules, we now consider the parsing of a conditional rule such as "if you see a six, write a one". Using "S" to as the symbol for **SEE**, we can present this to the model as "S 6 W 1". We then add actions to the basal ganglia such that a `phrase` of **VERB⊛SEE** will cause that `phrase` to be routed to a new `condition` group of neurons. A global `state` population is created, which gets inputs from all cortical areas that could be used as part of a `condition` (in this case, just `vision`), so that if `vision` is **TWO** then the value **TWO⊛SEE** will be added to `state`. The `state` and `condition` vectors are then compared (by computing the dot product) in a `similarity` population. Finally, the `go/nogo` population uses the `similarity` and `condition` values to compute a penalty to be applied to the utilities of actions in the basal ganglia. That is, it decreases the utility if there is a `condition` but `condition` does not match the current `state`.

Importantly, once the `condition` is stored in a separate neural population, we can now combine the `condition` and the `phrase` into a single semantic pointer. For this case, the resulting vector would be **CONDITION⊛(SEE⊛SIX) +VERB⊛WRITE+NOUN⊛ONE**. This is a single vector representing a complex, syntactically structured command that can be successfully executed by this model. In (Choo & Eliasmith, submitted), we develop a model capable of following a collection of rules of this form, but the model presented here is the first biologically realistic spiking model capable of taking the sequential input "S 6 W 1" and parsing it to create the correct semantic pointer vector.

Interestingly, the model supports some syntactic variation, such as "6 S 1 W" or "6 S W 1". However, it will not perform correctly when one phrase is embedded in the center of the other ("S W 6 1", for example). This difficulty with center embedding is a well-studied feature of natural languages, and appears naturally in this model from the processing available to neural populations.

Finally, in order to successfully respond to a condition instruction in the other order ("W 1 S 6"), we must also add an extra action rule which stores the initial phrase (**VERB⊛WRITE+NOUN⊛ONE**) in `memory` before processing the conditional phrase. This extra cognitive load indicates this model finds it easier to process "If you see a six, write a one" than "Write a one if you see a six".

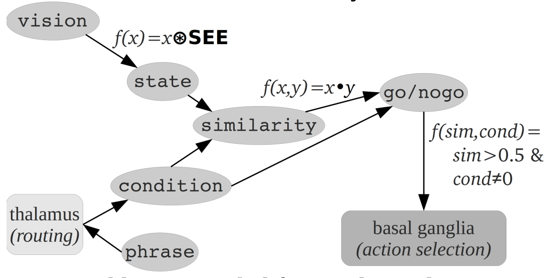

Figure 8: Additions needed for conditional instructions.

## Conclusion

We have shown how a model consisting of spiking leaky-integrate-and-fire neurons with properties and connection patterns that match the human brain can take a visually presented input command, parse it, and perform the correct action. This works for simple verb-noun commands and for more complex conditional commands, and also scales up to a vocabulary size of hundreds of thousands of terms. The majority of the neural components are identical those in our previous models (e.g. Eliasmith et al., 2012).

To perform this parsing, the model builds a single combined vector representation of the command. This resulting structured representation is of exactly the same form as those we have used in other neural models. As such, this model provides a potential explanation as to how brains can form and manipulate these symbol-like structures that are found throughout cognition.

That said, the current model has many limitations. It does not impose particular grammatical rules (other than avoiding center embedding). Perhaps relatedly, it does not address the problem of ambiguous classifications. It also does not perform token separation, as it requires that the input be already sequentially arranged. Fortunately, these problems have been addressed by other researchers, and our ongoing work is to adapt their solutions to the constraints of a biologically realistic spiking model. In particular, Ball (2011) provides an extensive project to process natural language speech using the ACT-R cognitive architecture, which may be adaptable to our neural framework.

## References

Ball, J. (2011). A Pseudo-Deterministic Model of Human Language Processing. *33rd Cog. Sci. Society Conference.*

Choo, X and Eliasmith, C. (submitted). General Instruction Following in a Large-Scale Biologically Plausible Brain Model. *35th Cog. Sci. Society Conference.*

Eliasmith, C. (2013). *How to build a brain.* Oxford University Press, New York, NY.

Eliasmith, C. & Anderson, C. (2003). *Neural Engineering.* Cambridge: MIT Press.Eliasmith et al., 2012

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

Hinton, G.E. (2010). Learning to represent visual input. *Phil. Trans. Roy. Soc. B*, 365, 177-184.

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

Singh, R., & Eliasmith, C. (2006). Higher-dimensional neurons explain the tuning and dynamics of working memory cells. J*ournal of Neuroscience*, 26, 3667-3678.

Stewart, T.C., Choo, X., and Eliasmith, C. (2010). Dynamic Behaviour of a Spiking Model of Action Selection in the Basal Ganglia. *10th Int. Conf. on Cognitive Modeling.*

Stewart, T. C., Tang Y., & Eliasmith C. (2011). A Biologically Realistic Cleanup Memory: Autoassociation in Spiking Neurons. *Cog.Systems Research*. 12(2), 84-92.