Spaun: A Perception-Cognition-Action Model Using Spiking Neurons

Terrence C. Stewart (tcstewar@uwaterloo.ca)
Feng-Xuan Choo (fchoo@uwaterloo.ca)
Chris Eliasmith (celiasmith@uwaterloo.ca)
Centre for Theoretical Neuroscience, University of Waterloo,
Waterloo, ON, N2L 3G1

Abstract

We present a large-scale cognitive neural model called Spaun (Semantic Pointer Architecture: Unified Network), and show simulation results on 6 tasks (digit recognition, tracing from memory, serial working memory, question answering, addition by counting, and symbolic pattern completion). The model consists of 2.3 million spiking neurons whose neural properties, organization, and connectivity match that of the mammalian brain. Input consists of images of handwritten and typed numbers and symbols, and output is the motion of a 2 degree-of-freedom arm that writes the model's responses. Tasks can be presented in any order, with no "rewiring" of the brain for each task. Instead, the model is capable of internal cognitive control (via the basal ganglia), selectively routing information throughout the brain and recruiting different cortical components as needed for each task.

Keywords: Neural engineering; cognitive architecture; spiking neurons; cognitive control; whole-brain systems

Introduction

In a forthcoming book, Eliasmith (2012) details a neural architecture for biological cognition called the *semantic pointer architecture* (SPA). This architecture, based on the Neural Engineering Framework (Eliasmith & Anderson, 2003), uses groups of spiking neurons to form distributed representations of high-dimensional vectors, which can in turn encode symbol-like tree structures. Synaptic connections between groups of neurons compute particular functions on those vectors, allowing high-level cognitive algorithms to be implemented in detailed spiking neuron models.

In this paper, we present an overview of the Semantic Pointer Architecture: Unified Network (Spaun) model and discuss its behaviour on six different tasks. We demonstrate that this biologically plausible spiking neuron model has the following features:

Task Flexibility: No changes are made to the model between tasks. Visual input indicates which task to do next.

Motor Plans: Model output provides a motor plan for a simple 2-joint arm, giving hand-written digits as responses.

Visual Memory: Even after an input has been recognized and classified as a particular symbol, details of the original image can still be recovered and used.

Compositionality: Multiple items can be represented and reliably bound together, allowing for the creation and manipulation of symbol-tree-like structures.

Symbolic Induction: Language-like patterns in visual input can be discovered after only a few presentations, and used to guide subsequent responses.

Input to the model consists of idealized and hand-written digits and symbols (Figure 1a). These images are given as a 28x28 grid of pixels. This input domain has the advantage of including significantly variable, real-world input, while also providing a reasonably limited semantics that the model must reason about. Output from the model consists of the motion of a 2-degree-of-freedom arm. The neural model generates a sequence of target locations which directly drive the controller for the arm. This provides the model with its own handwriting output (Figure 1b).

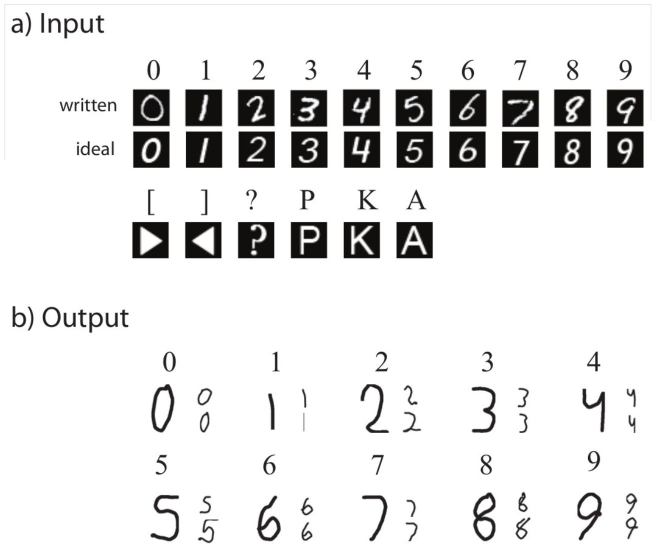


Figure 1: Example visual input and arm-movement output from Spaun. Input digits are from the MNIST database, and input symbols are used to inform the model of details of the current task. The model produces output by controlling a 2-joint arm, and variation in the internal representations produces the variation in its output hand-writing.

We can think of Spaun as having a single, fixed eye and a single 2-joint arm. The eye does not move, but instead the experimenter changes the image falling on it by showing different inputs over time, with each input shown for 150ms, followed by 150ms of blank background. To begin a specific task, Spaun is shown the letter "A" followed by a number between zero and seven. The subsequent input is then interpreted by the model in the context of the specified task and processed accordingly, resulting in arm movements that provide Spaun's response. All internal processing is performed using spiking neurons, with neural properties and connectivity consistent with the mammalian brain.

Neural Engineering Framework

While complete details on the construction of Spaun are available elsewhere (Eliasmith, 2012; <http://nengo.ca>), it is based on our continuing work on the Neural Engineering Framework (NEF; Eliasmith & Anderson, 2003). The NEF is a generic method for converting high-level algorithms into realistic spiking neuron models.

Two basic principles of the NEF are that a) groups of neurons form distributed representations of vectors, and b) connections between groups of neurons specify a computation to be performed on those vectors. Importantly, the NEF provides a method for analytically solving for the synaptic connection weights that will efficiently compute any given function.

While the NEF supports any type of neural model, for Spaun we use Leaky Integrate-and-Fire (LIF) neurons. The various properties of the LIF model (refractory period, capacitance, resistance, post-synaptic time constant, etc.) are set to be consistent with known neurophsyiological results for the various brain regions modelled.

To represent a vector using a group of neurons, the NEF generalizes the idea of preferred direction vectors. Each neuron in a group has a randomly chosen vector ***e*** for which it will fire most strongly. In particular, the amount of current J flowing into the neuron is the dot product of the preferred vector ***e*** with the represented value ***x***, times the neuron's gain α, plus the background current J_{bias} (Eq. 1).

While Eq. 1 lets us convert ***x*** into neural activity, we can also do the opposite by computing ***d*** via Eq. 2. This produces a set of linear decoding weights that can be multiplied by the activity of each neuron in the group. The result is the optimal least-squares linear estimate of ***x***. Thus, given a spiking pattern we can estimate what value is currently represented by those neurons.

Most crucially, we can use ***d*** to calculate the synaptic connection weights that will compute particular operations. To compute a linear operation where one group of neurons represents ***x*** and a second group should represent ***Mx***, where ***M*** is an arbitrary matrix, we set the connection weights between neuron i in the first group and neuron j in the second group to ω_{ij} as per Eq. 3. For non-linear operations, we need to compute a new set of ***d*** values via Eq. 4.

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

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

$$\omega_{ij} = \alpha_j \boldsymbol{e}_j \boldsymbol{M} \boldsymbol{d}_i \tag{3}$$

$$\boldsymbol{d}^{f(\boldsymbol{x})} = \Gamma^{-1}\Upsilon \quad \Gamma_{ij} = \int a_i a_j \, dx \quad \Upsilon_j = \int a_j f(\boldsymbol{x}) \, dx \tag{4}$$

This approach allows us to convert a high-level algorithm written in terms of vectors and computations on those vectors into a detailed spiking neuron model. Importantly, this approach works for recurrent connections as well. For example, we can implement memory by connecting a group of neurons back to itself with connections weights determined by Eq. 3 where ***M*** is the identity matrix. If that group of neurons is currently representing ***x***, then given no external input it will drive itself to keep representing ***x***, thus storing information over time.

Spaun

The Semantic Pointer Architecture: Unified Network model consists of multiple modules, depicted in Figure 2. These modules are considered to be cortical and subcortical areas that implement different operations. All components consist of LIF neurons connected via synaptic weights (Eq. 3), but each area computes a different set of functions.

To perform a particular task, information must be selectively routed between cortical areas, as each task uses a different subset of the components. This is achieved through an action selection system modelled after the mammalian basal ganglia and thalamus. We have previously shown that this model matches the anatomy and timing behaviour of the basal ganglia (Stewart, Choo, & Eliasmith, 2010) and provides enough flexibility to perform planning and problem solving in our model of the Tower of Hanoi task (Stewart & Eliasmith, 2011).

The model presented here is the first use of this neural action selection system with multiple tasks and detailed perceptual-motor systems. The action selection system is a neural production system, allowing us to write rules of the form "if cortical area X_1 matches the vector ***a*** and cortical area X_2 matches the vector ***b***, then send vector ***c*** to area X_3 and route the vector from area X_4 to area X_5". These rules are implemented by converting the rules into functions, applying Eq. 3, and using the resulting synaptic connection weights between the cortex, basal ganglia, and thalamus.

Importantly, the set of rules is fixed across all the tasks, giving a single, unified model. All input to Spaun is through its perceptual system, and all behavioral output from its motor system. The representational repertoire, background knowledge, cognitive mechanisms, neural mechanisms, etc. remain untouched while the system performs any of the tasks in any order.

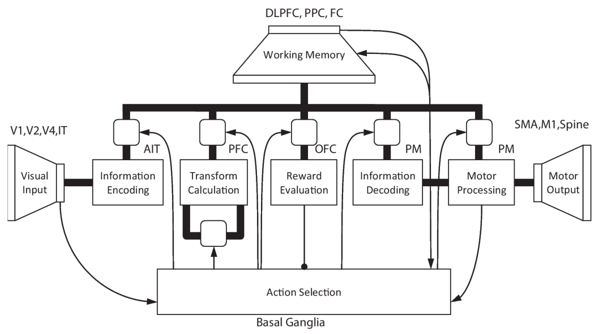


Figure 2: The Spaun architecture

The Spaun model (Figure 2) consists of three hierarchies, an action selection mechanism, and five subsystems. The first hierarchy is the visual system, which compresses an input image into a high-level abstract representation of that input. We adapt Hinton's (2010) Deep Belief Network to use LIF spiking neurons (via the NEF) and use it to compress a 28x28 image into a 50-dimensional vector we refer to as a *semantic pointer*: it is semantic because the high-level representation maintains similarity relationships from the image space; and it is a pointer because, as we will see, the system can recover the original information from the compressed form. Similarly, Spaun includes a motor hierarchy which dereferences an output semantic pointer representing a number into a motor plan to drive a two degree-of-freedom arm (DeWolf, 2010). A third internal hierarchy (discussed in more detail below in the *serial working memory* section) forms a working memory capable of binding and unbinding arbitrary semantic pointers, providing the compositionality that is crucial for complex cognition. The working memory component also provides stable representations of intermediate task states, task subgoals, and context. Anatomically, these functions cover large portions of prefrontal and parietal cortex.

The five subsystems, from left to right in Figure 2, are used to: 1) map the visual hierarchy output to a conceptual representation as needed (**information encoding**); 2) extract relations between input elements (**transformation calculation**); 3) evaluate the reward associated with the input (**reward evaluation**); 4) map output items to a motor semantic pointer (**information decoding**); and 5) control motor timing (**motor processing**). Several of the subsystems and hierarchies consist of multiple components needed to perform the identified functions. For instance, the working memory subsystem includes eight distinct memory systems, each of which can store semantic pointers. Overall, the model uses 2,341,212 spiking leaky integrate-and-fire (LIF) neurons. Additional details necessary to fully re-implement the model, and a downloadable version of the model can be found at <http://nengo.ca>.

Digit recognition

The simplest task for Spaun is digit recognition. We present the input sequence **A1[X?**, where **X** is a randomly chosen hand-written digit from the MNIST database. Each symbol is shown for 150ms, with 150ms between symbols. To perform this task, Spaun includes synaptic connections in the action selection system to implement the following algorithm:

- If visual input matches **A**, store **?** in the *state* area of the working memory. (This tells the system it is about to start a new task.)
- If **?** matches *state*, route the output of the visual encoding system to the state area of working memory. (This identifies and remembers which task is to performed.)
- If the visual input is **[**, activate the routing between the information encoding system and the general-purpose working memory. (This tells Spaum to store whatever digits appear next.)
- If the visual input is **?** and *state* is 1, route the pattern in working memory to the motor system.

Responses from this model for different inputs X are shown in Figure 3. Recognition accuracy is 94%, which compares well to humans on this task (~98%; Chaaban and Scheessele, 2007). The model is thus capable of correctly categorizing over a wide range of input variability.

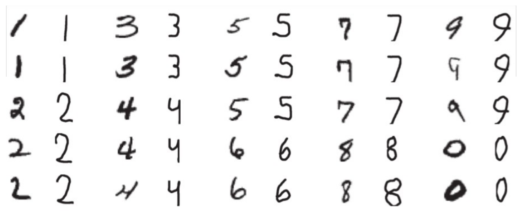


Figure 3: Input-output pairs for 20 different inputs. Each input (on the left) is correctly recognized by the model, which produces the output (on the right) via motor control.

Tracing from memory

While the previous task showed that the model can treat very different stimuli as tokens of the same type, we also want the model to be able to be sensitive to the variations within a type. To demonstrate this ability, we ask the model to do digit recognition, but to draw its response *in the same style as the original input.*

This is implemented by defining, for each digit, five different motor control sequences that draw five visually distinct versions of that digit. If the input pattern is X and the motor sequence is Y, then we can define the tracing function f(X)=Y. We build a neural connection between the vision system and the motor system via Eq. 3, and allow it to be selectively controlled via the action selection in the basal ganglia. If we now present new input X_{new} that was not among the 5 original inputs, the resulting Y_{new} value will be the model's linear extrapolation of what motor sequence would be appropriate for that novel input pattern.

To use this system, we use the sequence **A0[X?** and add a single rule to the basal ganglia action selection system:

- If the visual input is **?** and *state* is 0, route the pattern in working memory to the motor system via the tracing function.

This rule, combined with the previous ones for digit recognition, result in the behaviour shown in Figure 4. Note that it is capable of drawing 2's both with and without loops, 6's where the loops join in different locations, and generally following the slanting of the digits. It is not a perfect reconstruction of the original input, but it does demonstrate that while the internal neural representation of all 2's are very similar (as shown by Spaun's success in the previous task), there are still variations in the representation due to different visual features, and those variations can be used to successfully drive behaviour.

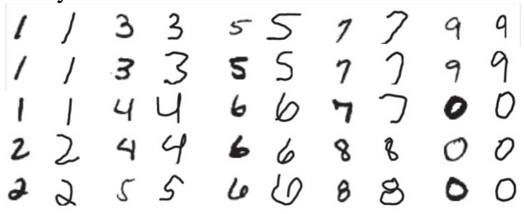


Figure 4: Input-output pairs for tracing from memory. The inputs (on the left) are recognized by the model, and then recreated from memory based on representational similarity to five previously known example pairs for each digit.

Serial Working Memory

For this task, Spaun is given a list of numbers and must repeat them back, in order. The algorithm used here is based on our previous work (Choo & Eliasmith, 2010) on a special-purpose serial memory model.

The NEF gives us a method for representing vectors via spiking neurons. The Semantic Pointer Architecture approach maps symbols onto particular vectors, so one vector might represent ONE while another vector represents TWO. Each vector, when represented, would produce a distinct firing pattern in that population of neurons.

To represent an ordered sequence, we cannot simply add the vectors together, since we could not distinguish ONE+TWO from TWO+ONE. Instead, we can perform *binding* by creating a new vector from two different vectors. We start by introducing new vectors for different positions: P1, P2, P3, etc. We then store the sequence ONE, TWO by representing the vector ONE$\otimes$P1+TWO$\otimes$P2. Many mathematical operations can be used for $\otimes$; we choose circular convolution since it is easy to accurately implement in spiking neurons using the NEF. This approach to representation using vectors is generally known as a Vector Symbolic Architecture (Gayler, 2003), and has been shown to scale well to adult-level vocabulary and grammar.

The action selection rules for this task are based on the previous one, with the addition of the **]** marker to indicate the end of a sequence. Figure 5 shows not only the input and output of the model, but also the ongoing spiking behaviour in various areas during the execution of the task.

Figure 5 also shows a method for interpreting what is currently being represented in a particular cortical area. The second working memory line shows how similar the current pattern in the working memory is to various different ideal patterns[1]. In particular, after the presentation of the final digit (t=2 seconds), the value being represented is similar to FOUR$\otimes$P1, THREE$\otimes$P2, TWO$\otimes$P3, and SIX$\otimes$P4. In other words, this one group of neurons is capable of storing any arbitrary sequence of digits. As the sequence gets longer, accuracy decreases, and both primacy and recency effects are seen (Choo & Eliasmith, 2010).

Question Answering

In addition to simply repeating a sequence, Spaun is capable of answering questions about the sequence. In particular, it can identify which digit is at a given location, and it can identify the location of a given digit.

This is accomplished by adjusting the transformation which takes the contents of working memory and routes it to the motor area. Given the vector S=FIVE$\otimes$P1+SIX$\otimes$P2, we can find the digit in position 1 by computing S$\oslash$P1, where $\oslash$ is circular correlation, since S$\oslash$P1$\approx$FIVE. The accuracy of this approximation is dependent on the length of the sequence, the number of neurons used, and the dimensionality of the vectors.

[1] Formally, this is the dot product between the ideal vector for that symbol and the decoded value found using Eq. 2.

To implement this task, the sequence is presented in the same manner as the serial working memory task. We then present the symbol P for a query based on position, or the symbol K for a query based on the kind of digit to look for in the list. The action selection rules route the appropriate transformation vector to the working memory area, providing the required information to the motor system.

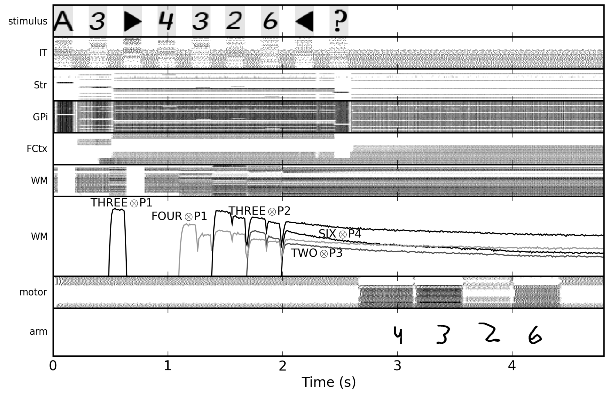
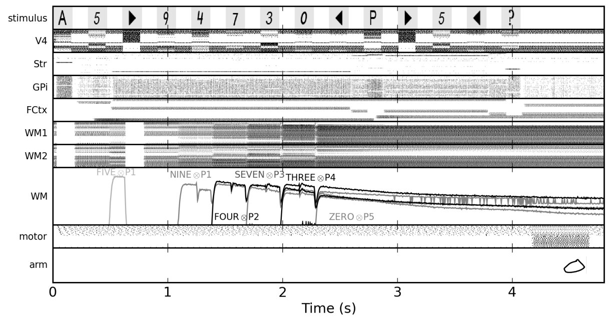


Figure 5: Serial recall of the sequence 4,3,2,6. Infero-temporal cortex (IT) holds the compressed representation of the visual input. Striatum (Str) activity determines how good a match each rule is to the current state. Globus pallidus internus (GPi) performs action selection, inhibiting all but the current best-matching rule. Frontal cortex (FCtx) holds task information, and working memory (WM) stores the list as a single vector: FOUR$\otimes$P1+THREE$\otimes$P2+TWO$\otimes$P3+SIX$\otimes$P4.



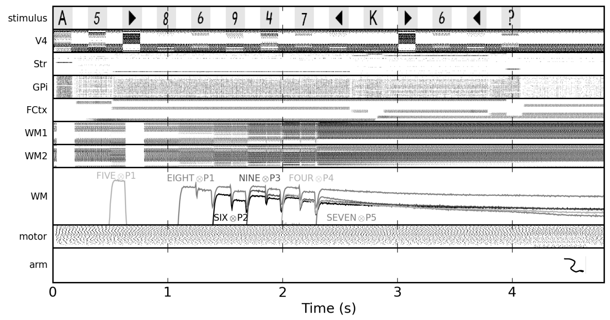


Figure 6: Answering questions about a list. The first case presents the list 9,4,7,3,0 and asks what is position 5. The model correctly answers 0. The second case presents the list 8,6,9,4,7 asks where the 6 can be found. The model correctly answers that it is in location 2.

Addition by Counting

For the fifth task, we show that Spaun is capable of performing sequences of internal actions, where there are a multiple steps to go through before producing a final output. This is demonstrated here by performing mental addition via counting. That is, to compute 4+3, the model must go through the steps of counting 4, 5, 6, and 7, entirely internally, and then finally producing the output 7.

Spaun achieves this by having multiple, general-purpose working memories. We use the first of these (WM1; the same neurons that stored the list and the recognized numbers in the previous task) to store the current value. A second group of neurons (WM2) stores the number of counting steps that are needed, and a third group (WM3) stores how many steps have been made.

Figure 7 shows Spaun performing this task over time for the specific case of 4+3. Importantly, the model produces accurate results for any single-digit addition. Furthermore, Spaun exhibits the expected linear relationship between subvocal counting and response times, as seen in human subjects (Cordes et al., 2001). That is, each counting step requires 419±10ms, which is within the empirical range of 344±135ms for subvocal counting (Landauer, 1962).

Successful counting demonstrates that flexible action selection is effectively incorporated into Spaun. It also shows that the model has an understanding of order relations over numbers, and can exploit that knowledge to produce appropriate responses.

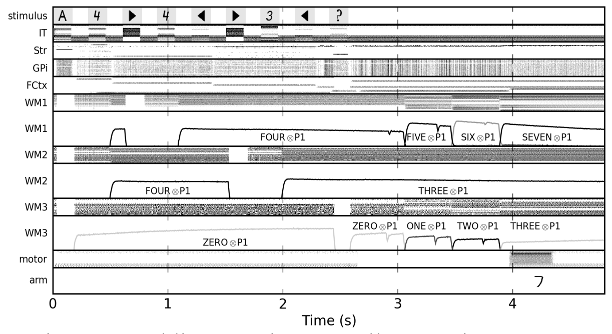


Figure 7: Adding 4+3 by mentally counting 4, 5, 6, 7 (WM1). WM2 keeps track of the fact that it should stop counting when it reaches the third step, and WM3 keeps track of what step it is at (0, 1, 2, 3).

Pattern Completion

Finally, we show that Spaun is capable of quickly identifying and responding to patterns in its input via inductive learning. This specifically targets a type of task that has long been held to be problematic for connectionist approaches: the ability to rapidly create and bind symbolic variables (e.g. Marcus, 2001; Jackendoff, 2002). The following example (from Hadley, 2009) shows a pattern completion task that humans are readily able to solve given only a few items. They are told that if they hear "biffle biffle rose zarple", the correct response is "rose zarple". After a three such examples, they must generalize to a new case:

Training Set

- Input: *Biffle biffle rose zarple.* Output: *rose zarple*.
- Input: *Biffle biffle frog zarple.* Output: *frog zarple*.
- Input: *Biffle biffle dog zarple*. Output: *dog zarple*.

Test Case

- Input: *Biffle biffle quoggie zarple*. Output: *?*

Hadley suggests that this task requires rapid variable creation because the second last item in the list can take on any form, but human cognizers can nevertheless identify the overall syntactic structure and identify "quoggie zarple" as the appropriate response. So it seems that a variable has been created, which can receive any particular content, and that will not disrupt generalization performance.

Figure 8 shows Spaun's behavior on a stimulus sequence with the same structure as that proposed by Hadley. Importantly, this is done extremely quickly (~2 seconds, consistent with human performance), and *without changing neural connection* weights. In other words, there is no learning rule; Spaun is able to learn to complete this pattern without any neural rewiring.

To achieve this result, we use a simplification of our earlier work with a neural model capable of performing Raven's Matrices (Rasmussen & Eliasmith, 2011). We store the representation of the first list in one area of working memory (WM1) and a representation of the second list in another area (WM2). Since we are using the semantic pointer method of representing lists, these lists are encoded as two high-dimensional vectors (V1 and V2). We can then compute the transformation T which takes the first vector (V2=V1⊗T, so T=V2⊘V1). We implement this is Spaun by adding a cortical area which computes the transformation between two working memory components, and adding rules to the basal ganglia to route this information appropriately when performing this task. As more examples are given, the value T is built up as the average over all the examples, improving accuracy.

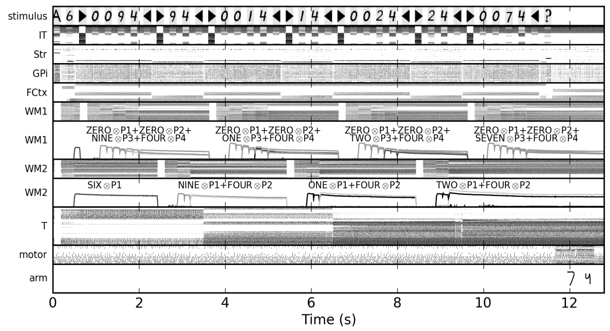


Figure 8: Pattern completion solving Hadley' rapid variable creation problem. The input consists of pairs of lists. After seeing 0094→94, 0014→14, and 0024→24, it correctly concludes that given 0074, it can complete the pattern by outputting 74.

Discussion

The basic components of the model presented here are not new; we have previously published spiking neuron models capable of exhibiting list memory (Choo & Eliasmith, 2010), pattern completion (Rasmussen & Eliasmith, 2011), action selection (Stewart, Choo, & Eliasmith, 2010), and sequential reasoning (Stewart & Eliasmith, 2011). However, the work presented here is the first demonstration of these capabilities in a single, unified model. There are no adjustments made to the model between tasks, and indeed the model can seamlessly go from one task to the next.

Furthermore, we feel that an important feature of this cognitive model is that it includes the entire system: visual perception, cognition, and motor action. The neural representations used throughout the model are the same, as are the underlying computational principles, and methods of mapping to neural spikes. As a result, this single model has neuron responses in visual areas that match known visual responses, as well as neuron responses and circuitry in basal ganglia that match known responses and anatomical properties of basal ganglia, as well as behaviorally accurate working memory limitations, as well as the ability to perform human like induction, and so on. Spaun is thus both physically and conceptually unified.

It should be noted that the current set of tasks Spaun can perform are in a constrained semantic space – that of lists of numbers. However, the basic principle of using high-dimensional vectors that can be bound together (i.e. semantic pointers) generalizes to more complex domains.

Furthermore, the model's architecture is not tightly tied to the set of tasks being implemented. That is, rather than having particular components to perform each task, the components presented here provide generic cognitive capacities, and any given task can recruit these components as needed. For example, the Pattern Completion task requires use of a component that can find the transformation that relates the information in two different areas of working memory. This cortical component would also be useful for performing other tasks, such as a Raven's Matrix task (Rasmussen & Eliasmith, 2011).

For the model presented here, all synaptic connection weights between neurons are analytically derived, rather than having them be learned, as in traditional neural network models. While this demonstrates that our model is capable of learning without connection weight changes (as in the pattern completion task), it leaves open the question of how these connections are learned in the real brain. While we do not have a complete developmental story for the various cortical components, we have developed a dopamine-based reinforcement learning system (Stewart, Bekolay, & Eliasmith, 2012) that has been integrated with Spaun in an n-arm bandit task, but the results are not presented here due to space limitations. This system allows Spaun to learn the connections between the cortical components and the basal ganglia, allowing the model to learn to recruit different components for different tasks.

Spaun presents a detailed spiking neural model capable of visual recognition, cognitive control, working memory, symbolic manipulation, and producing hand-written motor outputs. This sort of model is required for connecting high-level cognitive theory and behavioural data to the biological constraints available from neuroscience.

References

Chaaban, I., & Scheessele, M. R. (2007). Human performance on the USPS database. *Technical Report,* Indiana University South Bend.

Choo, F., Eliasmith, C. (2010). A Spiking Neuron Model of Serial-Order Recall. In Richard Cattrambone & Stellan Ohlsson (Eds.), *32nd Annual Conference of the Cognitive Science Society*. Portland, OR: Cognitive Science Society.

Cordes, S., Gelman, R., Gallistel, C. R., & Whalen, J. (2001). Variability signatures distinguish verbal from nonverbal counting for both large and small numbers. *Psychonomic Bulletin & Review*, *8*(4), 698–707.

DeWolf, T. (2010). *NOCH: A framework for biologically plausible models of neural motor control*. Masters Thesis. University of Waterloo, Waterloo.

Eliasmith, C. (2012). *How to build a brain: A neural architecture for biological cognition.* Oxford University Press, New York, NY.

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

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.

Hadley, R. F. (2009). The problem of rapid variable creation. *Neural computation, 21*(2), 510–32.

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

Jackendoff, R. (2002). *Foundations of language: Brain, meaning, grammar, evolution.* Oxford University Press.

Landauer, T. (1962). Rate of implicit speech. *Perceptual and Motor Skills*, 1, 646.

Marcus, G. F. (2001). *The algebraic mind.* MIT Press, Cambridge, MA.

Rasmussen, D., Eliasmith, C. (2011). A neural model of rule generation in inductive reasoning. *Topics in Cognitive Science, 3*(1), 140-153.

Stewart, T.C., Bekolay, T., Eliasmith, C. (2012). Learning to select actions with spiking neurons in the basal ganglia. *Frontiers in Decision Neuroscience.* 6.

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., Eliasmith, C. (2011). Neural Cognitive Modelling: A Biologically Constrained Spiking Neuron Model of the Tower of Hanoi Task. In L. Carlson, C. Haelscher, & T. Shipley (Eds.), *33rd Annual Conference of the Cognitive Science Society*.