CENTER FOR
Brains
Minds+
Machines

CBMM Memo No. 142 **June 27, 2023**

Skip Connections Increase the Capacity of Associative Memories in Variable Binding Mechanisms

Yi Xie*[,1], Yichen Li*[,2] and Akshay Rangamani[1]
1: Center for Brains, Minds, and Machines, Massachusetts Institute of Technology
2: Department of Psychology, Harvard University

Abstract

The flexibility of intelligent behavior is fundamentally attributed to the ability to separate and assign structural information from content in sensory inputs. Variable binding is the atomic computation that underlies this ability. In this work, we investigate the implementation of variable binding via pointers of assemblies of neurons, which are sets of excitatory neurons that fire together. The Assembly Calculus is a framework that describes a set of operations to create and modify assemblies of neurons. We focus on the `project` (which creates assemblies) and `reciprocal-project` (which performs variable binding) operations and study the capacity of networks in terms of the number of assemblies that can be reliably created and retrieved. We find that assembly calculus networks implemented through Hebbian plasticity resemble associative memories in their structure and behavior. However, for networks with N neurons per brain area, the capacity of variable binding networks ($0.01N$) is an order of magnitude lower than the capacity of assembly creation networks ($0.22N$). To alleviate this drop in capacity, we propose a *skip connection* between the input and variable assembly, which boosts the capacity to a similar order of magnitude ($0.1N$) as the `Project` operation, while maintain its biological plausibility.

This material is based upon work supported by the Center for Brains, Minds and Machines (CBMM), funded by NSF STC award CCF-1231216.

Skip Connections Increase the Capacity of Associative Memories in Variable Binding Mechanisms

Yi Xie [1*], Yichen Li [2 *], Akshay Rangamani [1]

1 Center for Brains, Minds and Machines, Massachusetts Institute of Technology
2 Department of Psychology, Harvard University

Abstract

The flexibility of intelligent behavior is fundamentally attributed to the ability to separate and assign structural information from content in sensory inputs. Variable binding is the atomic computation that underlies this ability. In this work, we investigate the implementation of variable binding via pointers of assemblies of neurons, which are sets of excitatory neurons that fire together. The Assembly Calculus [1] is a framework that describes a set of operations to create and modify assemblies of neurons. We focus on the `project` (which creates assemblies) and `reciprocal-project` (which performs variable binding) operations and study the capacity of networks in terms of the number of assemblies that can be reliably created and retrieved. We find that assembly calculus networks implemented through Hebbian plasticity resemble associative memories in their structure and behavior. However, for networks with N neurons per brain area, the capacity of variable binding networks $(0.01N)$ is an order of magnitude lower than the capacity of assembly creation networks $(0.22N)$. To alleviate this drop in capacity, we propose a *skip connection* between the input and variable assembly, which boosts the capacity to a similar order of magnitude $(0.1N)$ as the `Project` operation, while maintaining its biological plausibility. [†]

1 Introduction

The human brain is one of the most sophisticated learning machines in the world, and consequently computational models of the brain have been of great interest to the learning community. There have been many models proposed to understand the brain, at different levels of abstraction, ranging from molecular models of neurotransmission and models of single neurons to whole brain models studied in cognitive science. Despite significant progress in experimental and theoretical computational neuroscience, understanding how molecules, cells, and synapses contribute to cognition, behavior, intelligence, reasoning, and language remains elusive. We are still searching for "a logic for the transformation of neural activity into thought and action" [2].

With this motivation, we turn to the study of assemblies (or ensembles) of neurons. Assemblies are large densely interconnected sets of neurons whose simultaneous excitation is tantamount to the subject's thinking of a particular concept or idea. They are initially created to record memories of external stimuli and are believed to be subsequently manipulated through a repertoire of operations in the non-sensory brain. Known as "the alphabet of the brain" [3], assemblies of neurons were first hypothesized seven decades ago by Donald O. Hebb [4] attempted to explain with his theory of plasticity, and were clearly identified as existing in mammalian brains through calcium imaging more than a decade ago [5,6]. Recently, the Assembly Calculus (AC) [1], a formal computational model of the brain based on assemblies of neurons, was introduced as a high-level framework that would qualify as the sought "logic" at a level of abstraction between models of neurons [7,8] and whole brain models. While modern deep neural networks are inspired by models of the sensory parts of the brain, especially the visual cortex [9], assemblies of neurons are meant to model

[*]Equal contributions.
[†]All code used to generate the figures in this work can be found at https://github.com/minzsiure/Variable-Binding-Capacity.

intermediate levels of computation beyond sensory information processing. The AC framework consists of operations that create new assemblies and modify existing ones, and has been shown to be Turing-complete [10]. Systems have also been written in assembly calculus to solve complex cognitive problems like language parsing [11] and planning in blocks world [12].

Assemblies of neurons are a promising candidate for the data structure underlying the brain's "logic", and thus it is worth investigating the basic operations of assembly calculus. In this paper, we focus on two fundamental operations in AC. The first operation is `project`, which creates assemblies from stimuli. The second operation is `reciprocal-project`, which implements *variable binding*. Variable binding is an atomic operation [13] that enables agents to separate and assign structural information from sensory inputs. It allows us to represent information in a structured manner to access entities and their attributes in different ways such as "What color is the block?" and "What object is blue?". While anatomical or convolutional approaches to variable binding exist, we study the implementation of pointer-based binding through `reciprocal-project`. Notably, the point-based binding operation has been shown to be critical to language parsing [11].

In this paper, we establish the above two AC operations, `project` and `reciprocal-project`, as *associative memories*. Since Hebbian plasticity is one of the computational primitives that underlies the implementation of operations in assembly calculus, one can expect that associative memory structures emerge in the neural circuits that implement them. We study the capacity of the associative memories that emerge in both these operations and find that the capacity of `reciprocal-project` is smaller than the capacity of `project` by an order of magnitude. We propose a biologically plausible skip connection in the architecture of `reciprocal-project` to mitigate this drop in capacity. This addition enables new opportunities for exploring hierarchical models using assemblies of neurons.

Our Contributions.

1. We establish that the neural circuits that implement operations for the creation of assemblies and variable binding in Assembly Calculus (`project` and `reciprocal-project`) display an associative memory structure.
2. We show that such associative memories yield a clear class structure after receiving multiple classes of stimulus, and that they can complete patterns based on partial or perturbed inputs.
3. We provide the measurement of the memory capacity of both `project` and `reciprocal-project` under different parameter settings (i.e., number of neurons per area, cap size, density of synaptic connections), and find that the capacity of variable binding (`reciprocal-project`) is much lower than the capacity of assembly creation.
4. We introduce a new variable binding model with an additional skip connection, and show that such a design boosts the memory capacity significantly while making the model more biologically plausible. This opens new avenues for exploring hierarchical models using assemblies of neurons.

Paper Outline. We introduce both the AC model and its basic operations in more detail in Section 2, as well as the relevant setup of the experiments. In Section 3, we establish that associative memories emerge in AC through Hebbian learning with a stream of stimuli drawn from some distribution; further, this phenomenon generalizes to multiple classes of stimuli. Having established that AC operations create associative memories, in Section 4, we measure the capacity of associative memories in AC as a function of model parameter N, the number of neurons in each brain area, and K, the maximum number of active neurons in an area at any time, and p, the rate of connectivity of brain areas and assemblies. We discuss the phenomenon of the cascading reduction in the capacity of assemblies over hierarchical brain areas. Then, we leverage our knowledge of `reciprocal-project`'s hypothesized biological role in variable binding mechanisms to propose the addition of *skip connection*. This addition allows a direct access to the pointer variable by the sensory input, which increases the capacity by an order of magnitude; as such, it tackles the challenges caused by the aforementioned phenomenon. Lastly, we discuss the *biologically plausiblility* of the addition of skip connections.

2 Assembly Calculus

In this section, we introduce the basic framework of Assembly Calculus and the two operations that we study: `project` and `reciprocal-project`.

2.1 Model Overview

The model of the brain used in Assembly Calculus consists of several brain areas each with N excitatory neurons. The neurons are binary (without any internal structure) and can either be firing or not firing (neuron set to 0 or 1). Computation occurs in discrete timesteps, and the activity in any brain area can be represented using a binary vector $\{0, 1\}^N$. Afferent connections between neurons in different areas are drawn independently at random with probability p. This means they have the structure of a random bipartite graph $B_{N,p}$. Recurrent connections between neurons in the same area follow the structure of an Erdos-Renyi graph $G_{N,p}$. While brains have excitatory as well as inhibitory neurons, AC only has excitatory neurons. The function of inhibitory neurons is instead implemented implicitly by a $\text{cap}_K(\cdot)$ operation, which selects the K neurons with the largest synaptic input out of the N neurons in a specific brain area to fire at a particular timestep. The parameter K is referred to as the cap size. The final key component in this model is the multiplicative Hebbian plasticity, which increases the strength of connection between neurons that fire in consecutive timesteps by a factor of β.

2.2 Stimuli: Representation of Distinct Concept Classes

A stimulus is defined by a pattern of activity - represented by a vector in $\{0, 1\}^N$. Stimuli are modeled as random vectors, where each neuron fires independently. Since the neural stimuli corresponding to different phenomena are going to have different representations, we introduce the idea of *concept classes* or *stimuli classes*. Each concept class is represented by a *coreset* of neurons that fires with a higher probability than the rest of the neurons in the stimulus area. The stimulus classes are thus fully specified by the coresets, the probability of firing within the coreset, and the background probability of firing. We denote these quantities by $\{S, r, q\}$ respectively. The coreset of stimulus class i is denoted by S_i, and we set its cardinality $|S_i| = K$. Every neuron that belongs to S_i fires with probability r, while the rest of the neurons fire with probability q which is typically much smaller than r. The parameters r, q can be thought of as determining the signal-to-noise ratio of this problem.

By manipulating $\{S, r, q\}$, we can create a diverse range of stimulus classes with varying levels of complexity, coreset activation, and noise. This flexibility allows us to simulate different scenarios and analyze how well our model can discriminate and classify various concept classes under different conditions. In this paper, we set $r = 0.9$ and $q = 0.01$. We choose the coresets uniformly at random from all K-sized subsets of N neurons.

2.3 `Project`

`Project` is the primary operation of AC that describes the creation of assemblies from stimuli. It entails that through the projection of a stimulus assembly x in area $\mathcal{S}$, an assembly y that can be thought of as a "copy" of x forms in a downstream area $\mathcal{C}$. Henceforth y fires every time x fires [1].

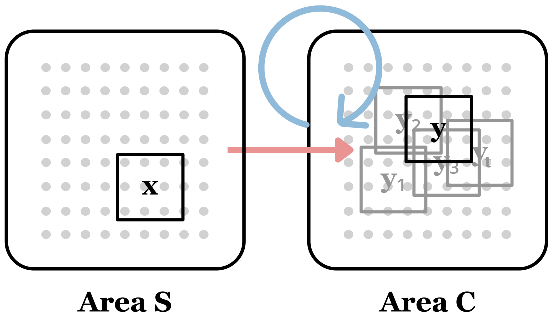

Figure 1: A demonstration of `project`$(x, \mathcal{C}, y)$, where an assembly x in stimulus area S is being projected to form an assembly y in a downstream assembly content area $\mathcal{C}$. Here, the red arrow represents the afferent synaptic connections $W_{\mathcal{SC}}$, and the blue arrow represents the recurrent synaptic connections $W_{\mathcal{CC}}$.

The idea is that, with repeated firing of x, afferent synaptic connectivity from area $\mathcal{S}$ to area $\mathcal{C}$ excites a sequence of $\{y^{(t)}\}$, sets of neurons of size K in area $\mathcal{C}$. With large enough parameters and high enough plasticity, this process converges exponentially fast with a high probability to create an assembly y as a result of the projection, as proven by previous literature [1]. A demonstration of this is shown in Figure 1.

More formally, if we consider a brain area $\mathcal{C}$ receiving synaptic input from an assembly x in stimulus area $\mathcal{S}$, we can write the dynamics of the projection operation as:

$$\begin{aligned} y^{(t+1)} &= \text{cap}_K \left(W_{\mathcal{CC}}^{(t)} y^{(t)} + W_{\mathcal{SC}}^{(t)} x \right) \\ W_{\mathcal{SC}}^{(t+1)} &= W_{\mathcal{SC}}^{(t)} + \beta y^{(t+1)} x^\top \odot W_{\mathcal{SC}}^{(t)} \\ W_{\mathcal{CC}}^{(t+1)} &= W_{\mathcal{CC}}^{(t)} + \beta y^{(t+1)} y^{(t)\top} \odot W_{\mathcal{CC}}^{(t)} \end{aligned} \tag{1}$$

In the above set of equations, $W_{\mathcal{SC}}$ and $W_{\mathcal{CC}}$ refer to the weights of the afferent and recurrent synaptic connections, respectively. $y^{(t)}$ refers to the pattern of activity in $\mathcal{C}$ at time t. At the end of T rounds of firing, an assembly y is created as the result of the operation `project`$(x, \mathcal{C}, y)$. In each round, the synaptic weights are updated according to the rule of Hebbian plasticity, by a factor of β. Theorem 1 of [14] shows that this process converges in the sense that after a certain number of timesteps no new neurons are activated in area $\mathcal{C}$.

In section 3, we show that the mechanism of `project` resembles the creation of associative memory: every time when the stimulus assembly x fires in area $\mathcal{S}$, assembly y in area $\mathcal{C}$ will also fire reliably, as a result of Hebbian updating. Assembly y is thus "associated" with x.

2.4 `Reciprocal-Project`

`Reciprocal-project` is an extension of `project` and is hypothesized to be instrumental for implementing variable binding in the brain [15,16]. In `reciprocal-project`, three brain areas are involved: a stimulus area $\mathcal{S}$, a content area $\mathcal{C}$, and a variable area $\mathcal{V}$. We denote the operation as `reciprocal-project`$(y, \mathcal{C}, z)$ where the stimulus x in $\mathcal{S}$ that created y fires to induce an assembly z in $\mathcal{V}$ that has strong forward and backward synaptic connectivity with y. The demonstration of this neural operation is shown in Figure 2

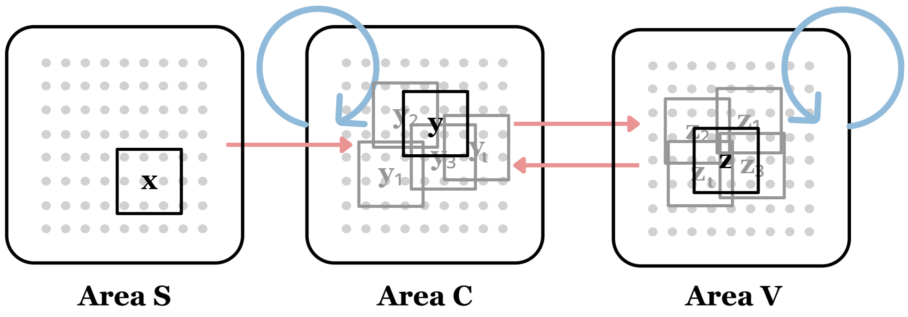

Figure 2: A demonstration of `reciprocal-project`$(y, \mathcal{C}, z)$, where an assembly x in stimulus area $\mathcal{S}$ is being projected to form an assembly y in a downstream content area $\mathcal{C}$, and with strong forward and backward synaptic connectivity to form an assembly z in the variable area $\mathcal{V}$.

Formally, with an additional variable region $\mathcal{V}$ involved for establishing strong backward synaptic connectivity as a variable binding mechanism, the dynamic of the `reciprocal-project`$(y, \mathcal{C}, z)$ operation is as follows:

$$\begin{aligned}
y^{(t+1)} &= \text{cap}_K\left(W^{(t)}_{\mathcal{CC}} y^{(t)} + W^{(t)}_{\mathcal{SC}} x + W^{(t)}_{\mathcal{V}C} z^{(t)}\right) \\
z^{(t+1)} &= \text{cap}_K\left(W^{(t)}_{\mathcal{CV}} y^{(t)} + W^{(t)}_{\mathcal{VV}} z^{(t)}\right) \\
W^{(t+1)}_{\mathcal{SC}} &= W^{(t)}_{\mathcal{SC}} + \beta y^{(t+1)} x^\top \odot W^{(t)}_{\mathcal{SC}} \\
W^{(t+1)}_{\mathcal{CC}} &= W^{(t)}_{\mathcal{CC}} + \beta y^{(t+1)} y^{(t)\top} \odot W^{(t)}_{\mathcal{CC}} \\
W^{(t+1)}_{\mathcal{CV}} &= W^{(t)}_{\mathcal{CV}} + \beta z^{(t+1)} y^{(t)\top} \odot W^{(t)}_{\mathcal{CV}} \\
W^{(t+1)}_{\mathcal{VC}} &= W^{(t)}_{\mathcal{VC}} + \beta y^{(t+1)} z^{(t)\top} \odot W^{(t)}_{\mathcal{VC}} \\
W^{(t+1)}_{\mathcal{VV}} &= W^{(t)}_{\mathcal{VV}} + \beta z^{(t+1)} z^{(t)\top} \odot W^{(t)}_{\mathcal{VV}}
\end{aligned} \tag{2}$$

The weights update is consistent with the rule of Hebbian plasticity, by a factor of β. In addition to the formed assembly y described in `project`, an additional assembly z is created in the variable area $\mathcal{V}$ as the result of the `reciprocal-project` operation.

3 The Emergence of Associative Memories in Assembly Calculus

In the above section, we introduced two of the fundamental operations of AC, `project` and `reciprocal-project`. While previous work has shown that these mechanisms converge to stable sets of assemblies, we aim to provide a description of the neural circuits in terms of the synaptic weight matrices. In this section, we study the patterns and structures that emerge in these neural circuits and describe them in terms of *associative memories*. We establish the correspondence between associative memories and models in assembly calculus in two ways. First through the spectral decomposition of the synaptic weight matrices. Second, by showing that circuits learned during AC operations can reconstruct assemblies from incomplete or noisy stimuli. We also show that these properties hold when the stimuli come from multiple stimulus classes.

3.1 Associative Memories

Associative memories have been a popular topic in neural networks for over half a century, starting with the work of Kohonen [17] who proposed a mathematical model for a non-hierarchical pattern storage system.

This work inspired many subsequent studies, including the Self-Organizing Map algorithm by Kohonen [17] and the Simple Recurrent Network by Anderson [18]. Hopfield [19] later proposed the Hopfield network, a recurrent neural network that can store and recall multiple patterns. Kanerva [20] proposed the sparse distributed memory, which uses high-dimensional binary vectors for efficient pattern storage. Associative memories have also been used in signal processing applications such as holography before their systematic study in the context of neural networks [21].

Associative memories are systems that store associations between stimuli and responses. Given a number of stimulus-response pairs $\{(x_i, y_i)\}_{i=1}^N$, an associative memory A when probed with a stimulus x_i will return the response y_i. In the special case that stimuli are associated with themselves, the associative memory is called *auto-associative*. When the stimuli and responses are different, we refer to the memory as *hetero-associative*. If the stimuli x_i are orthogonal or near orthogonal, we can construct an associative memory from the data as $A = \sum_{i=1}^N y_i x_i^\top = YX^\top$. We thus say that a matrix A has an associative memory structure if we can discover factors U, V such that $A = UV^\top$ and U, V are close to the outputs and inputs of A respectively. For an *auto-associative memory*, we have $A = XX^\top$ which means that we would like both factors U, V to be close to the inputs of A.

In addition to mapping stimuli to responses, associative memories are able to recall responses from noisy or incomplete patterns of the stimuli. An associative memory typically performs this function by implicitly constructing an energy function with local minima corresponding to the stored patterns. The iterative procedure that recovers the stored patterns in an associative memory can be viewed as an iterative minimization algorithm on the implicitly defined energy function. This characteristic of associative memory is commonly demonstrated by supplying the memory a fraction of an input stimulus - for instance, the left half of an image. The memory then reconstructs the complete image in the case of an auto-associative memory, or the corresponding response in the case of a hetero-associative memory.

3.2 Spectral Factors of Synaptic Connections in Assembly Calculus Resemble Associative Memories

We show that the synaptic connections created by Hebbian updates during AC operations exhibit an associative memory structure by decomposing the matrices into factors that correspond to the input and output assemblies. We discover these factors through the Singular Value Decomposition (SVD) of the afferent and recurrent connectivity matrices. We perform SVD on each of the weight matrices to obtain

$$W = U\Sigma V^T$$

where the columns of U contain left singular vectors, the rows of V^T contain right singular vectors, and the diagonal of Σ stores singular values (in descending order) of the weight matrix. We expect the left and right singular vector factors to contain information pertaining to the output and input assembly pairs, respectively. To quantify the similarity between a singular vector and an assembly, we compute the Hamming distance between them (with the top-k-winner-take-all operation applied to the singular vector to align with the representation of the assembly). A smaller distance signifies greater similarity or closeness between the two vectors. More precisely, we use the following distance metric between a right singular vector v and a stimulus x:

$$d_H(v, x) = \mathbf{1}^T \left|\text{cap}_k(|v|) - x\right|.$$

We can similarly compare a left singular vector u with the assembly y.

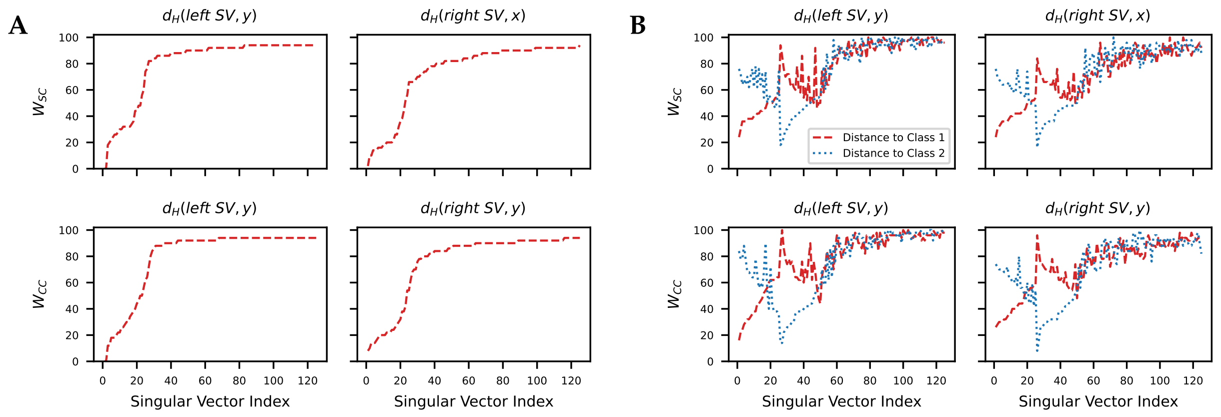

Figure 3: The visualization of Hamming distance between singular vectors in weight matrices of `project` and the input or output assembly. We normalize the weights of synaptic connections every 5 iteration. Here, $N = 1000$, $K = 50$, $p = 0.1$, $\beta = 0.1$. The model learns a class until reaching convergence with at most 30 iterations. We form assemblies at test time under 5 iterations when projecting stimuli generated under the setting described in section 2.2 using $q = 0.01$ and $r = 0.9$. (**A**) single class of stimuli. (**B**) binary classes of stimuli.

Using Figure 3 as an illustrative example, we present the Hamming distance between singular vectors in weight matrices of `project` and the input or output assembly. In Figure 3A, the first few left singular vectors in the afferent connectivity matrix $W_{\mathcal{SC}}$ are very close to the output assembly y in the content area $\mathcal{C}$, and the first few right singular vectors in $W_{\mathcal{SC}}$ are similar to the input vector x from the stimulus area $\mathcal{S}$. The rest of the singular vectors have a distance of $\approx 2K$ from the input stimulus and output assembly respectively, which means the singular vectors are completely orthogonal to the stimuli/assembly. The singular vectors in the recurrent connectivity matrix $W_{\mathcal{CC}}$ exhibit a comparable but less pronounced resemblance to the input and output assembly (in this case, both the input and output are the assemblies in area $\mathcal{C}$ itself). This indicates that the afferent connections form a hetero-associative memory, while the recurrent connections form an auto-associative memory.

A similar spectral structure emerges when projecting multiple classes of stimuli, as shown in Figure 3B. We project the first class of (noisy) stimuli to the network for 30 iterations, followed by projecting the second class of stimuli to the same network for another 30 iterations, with weight normalization after every 5 iterations. We observe an emerging class structure: different sets of singular vectors are close to the stimuli from different classes. The singular vectors that correspond to the first class are dissociated from those that correspond to the second class.

We repeat the SVD Hamming distance visualization for `reciprocal-project`, an operation employing three brain areas to facilitate variable binding. As depicted in Figure 4, the singular vectors in the learned weight matrices store associations of input and output pairs, and the structure of two stimuli classes is largely preserved. However, we observe that with the hierarchical structure of brain areas in `reciprocal-project`, such class structure becomes increasingly *noisier* in subsequent layers. The class structure is most distinct in the afferent connection $W_{\mathcal{SC}}$, which is situated at the top of the network. This suggests that as the stimuli signals traverse through multiple afferent and recurrent synaptic connections, the model can introduce additional noise due to the randomness in weight initialization. To address this issue, we explore the inclusion of a skip connection in `reciprocal-project` (see section 4).

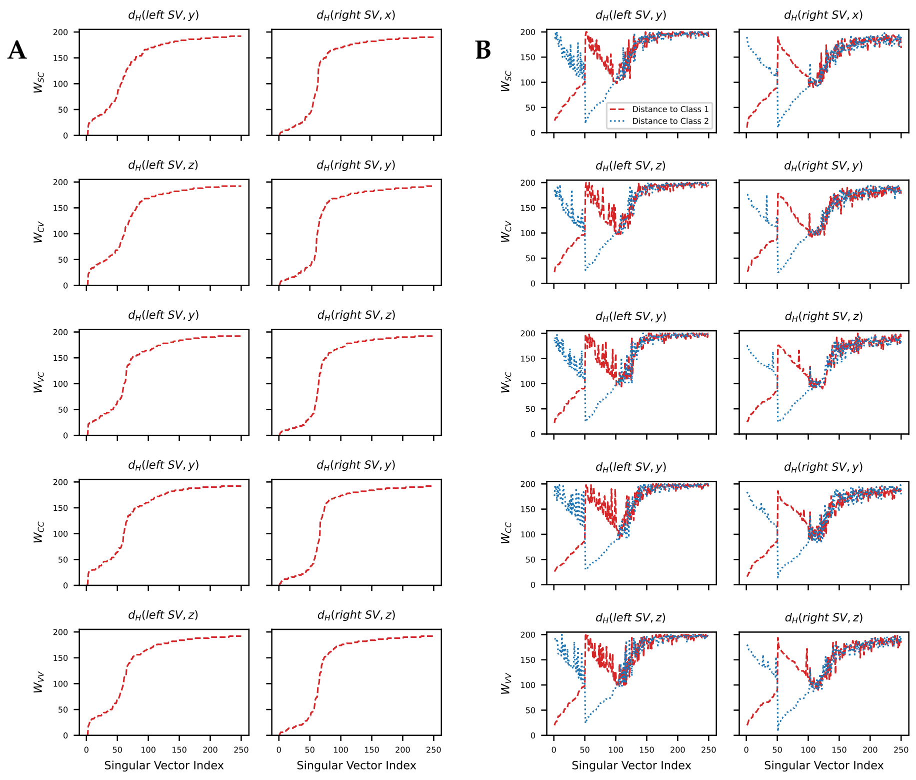

Figure 4: The visualization of Hamming distance between singular vectors in weight matrices of `reciprocal-project` and the input or output assembly. We normalize the weights of synaptic connections after every 5 iterations. Here, $N = 1000$, $K = 100$, $p = 0.1$, $\beta = 0.1$. The model learns a class until reaching convergence with at most 50 iterations. We form assemblies at test time under 5 iterations when projecting stimuli generated under the setting described in section 2.2 using $q = 0.01$ and $r = 0.9$. (**A**) single class of stimuli. (**B**) binary classes of stimuli.

In figures 3 and 4, we observe that the number of singular vectors which are closer than random to the stimuli/assemblies is $\approx K$. This is likely due to the cap operation in assembly calculus, which limits the rank of the updates to each synaptic weight matrix. However, when examining the multi-class stimulus experiments, we find that the number of singular vectors associated with each stimulus class can be significantly smaller than K. This implies the model has the capability to store multiple classes of stimuli-assembly associations, as we explore in the next section. We expect that the capacity of an assembly calculus circuit is limited by the rank of the learned synaptic weight matrix. Homeostasis (or normalization) plays a critical role here by preventing the rank of the weight matrices from collapsing (see appendix B for more details).

3.3 Assembly Calculus Circuits are Capable of Pattern Completion & Assembly Recall

In the previous section, we established that the weight matrices of AC operations have factors that correspond to the stimuli and assemblies. This means that the weight matrices have a structure similar to an associative memory. In this section, we demonstrate a key property of associative memories in AC operations - to recover stored patterns from noisy and incomplete stimuli [22]. The ability of AC circuits to reconstruct assemblies from a fraction of their neurons indicates that the recurrent synaptic weights form an associative memory. The ability of AC circuits to reconstruct assemblies from noisy input stimuli indicates that the afferent synaptic weights form an associative memory. Both demonstrations further support our claim that assembly calculus operations learn associative memories.

Assembly Pattern Completion: Previous work [1] has shown that assemblies can be recovered from a fraction of the individual neurons, provided that the original assembly has been adequately reinforced. We show that this is also the case for AC circuits that are learned from multiple classes of stimuli. We conduct an experiment on an AC network implementing `project`. We present stimuli from M different concept classes as described in section 2.2, each for T_{train} iterations to form assemblies $\mathbf{y}^m \in \{0,1\}^n, m = 1, \dots, M$ in area $\mathcal{C}$. For each class assembly, we make an incomplete version of $\mathbf{y}^m$ by retaining only a fraction $\alpha \in [0,1]$ of the K active neurons, thereby creating the subset assembly $\tilde{\mathbf{y}}^m$. We then attempt to recover $\mathbf{y}^m$ by setting $y_{(0)} = \tilde{\mathbf{y}}^m$ and employing the recurrent synaptic connections $W_{\mathcal{CC}}$ to execute the cap-K operation for T_{recover} steps. Mathematically, the neurons in brain content area $\mathcal{C}$ evolve as $y^m_{(t+1)} = \text{cap}_K \left(W_{\mathcal{CC}} y^m_{(t)} \right)$. We measure the performance of pattern completion by recording the percentage of the assembly recovered,

$$\frac{\left| S_{\mathbf{y}^m} \cap S_{y^m_{(T_{\text{recover}})}} \right|}{\left| S_{\mathbf{y}^m} \right|},$$

where S_y denotes the set of K winners in assembly y. The results of this experiment are shown in Figure 5, where we see that following sufficient training on multiple classes of stimuli (T_{train}), the model successfully completed assemblies from all classes with perfect accuracy in merely $T_{\text{recover}} = 2$ iterations. The parameter settings for this experiment are given in appendix C.

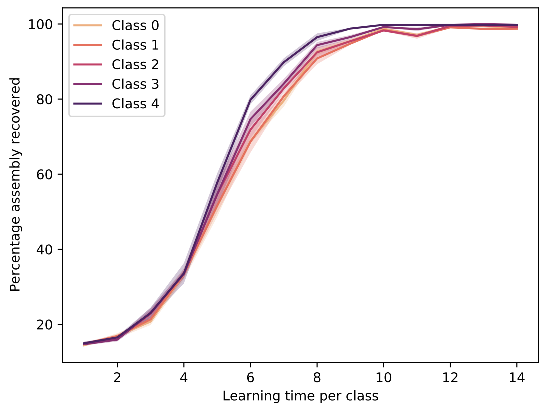

Figure 5: Pattern completion in `project`. The model forms assemblies from few-shot presentations of the stimulus classes and is able to recover the original assembly in $T_{\text{recover}} = 2$ rounds when probed with an incomplete assembly with only $\alpha = 50\%$ of its original K winners activated. The x-axis shows the number of examples (T_{train}) shown per class to form an assembly $\mathbf{y}^m$. We averaged the results over 5 trials and show both the median and standard error.

Assembly Recall: In the previous experiment we demonstrated that the recurrent synaptic connections $W_{\mathcal{CC}}$ in a `project` model form an associative memory. We now probe the model with corrupted inputs to show that it can recover the learned assemblies. This will demonstrate that the afferent synaptic connections $W_{\mathcal{SC}}$ also form an associative memory.

Similar to the pattern completion experiment, we present stimuli x^m from M different concept classes, each for T_{train} steps to form assemblies $y^m \in \{0,1\}^n, m = 1, \dots, M$ in area $\mathcal{C}$. We then probe the model with corrupted inputs $\tilde{x}^m$. The corrupted inputs can be created in two ways. First, we vary the probability of the coreset firing $r \in [0.1, 0.9]$. Alternatively, we create a corrupted input by perturbing its coreset to be a certain Hamming distance away from that of the original class coreset S, denoted $\tilde{S}$. The Hamming distance between the coresets varies from 0 to $2K$. Given the perturbed inputs $\tilde{x}^m$, we obtain its associated assembly $\tilde{y}^m$ by projecting $\tilde{x}^m$ for T_{recover} steps. The robustness of assembly recall was measured by the Hamming distance between the original assembly y^m and the recalled assembly $\tilde{y}^m$.

The results of the above experiments are shown in Figure 6. In both instances of perturbation, the model trained using `project` is able to reconstruct the original assembly within 10% of the original assembly y^m. This is true even when the distance between the original coreset and the perturbed coreset is close to 50%. In both instances of perturbation, the model trained using `project` is able to recall the original assembly at a decent rate even when the coreset firing is noisy or when the coreset is disturbed to be distant from the original stimuli. The robustness follows the expected trend, in which the reconstruction is closer to the original assembly when the signal-to-noise ratio of the stimuli is higher (figure 6A) or the magnitude of the perturbation is smaller (figure 6B).

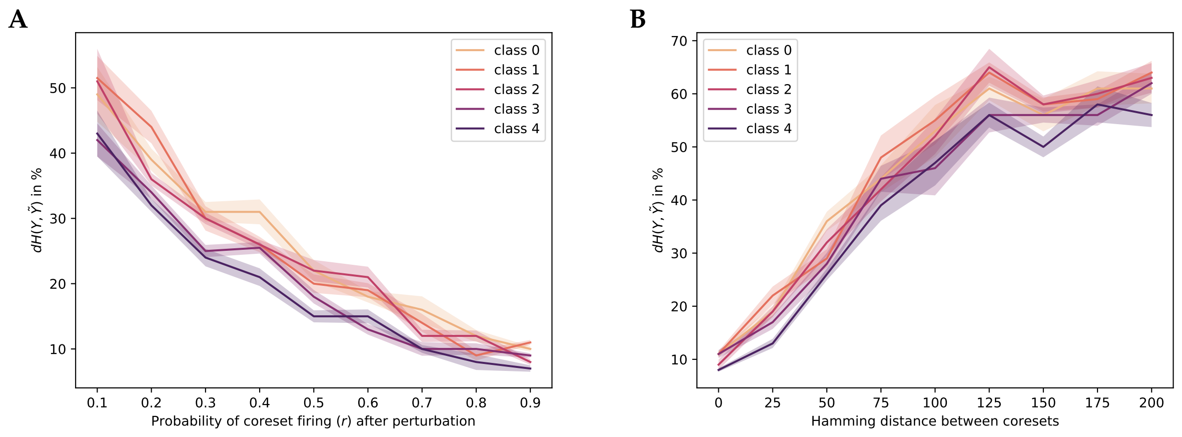

Figure 6: Assembly recall in `project`. Here shows the median and standard error of results across 5 trials. We denote a stack of stimuli x or assemblies y by X and Y. (**A**) Performance of recalling the original assembly Y (formed under stimuli X) across pertubations of r. We present the model a stream of corrupted inputs $\tilde{X}$ constructed by varying the coreset firing rate r in X. We plot $d_H(Y, \tilde{Y})$ normalized by K, the Hamming distance between the set of firing neurons of the original assemblies Y and of assemblies $\tilde{Y}$ formed by corrupted inputs $\tilde{X}$, and normalized by K to show the percentage of differences. (**B**) Performance of recalling the original assembly Y across manipulations of Hamming distance between coresets. Here, the corrupted inputs $\tilde{X}$ are drawn from coresets $\tilde{S}$ that differ by a certain Hamming distance from the coreset S of the original stimuli X. See more details of the experiment setup in appendix D.

In summary, in this section, we have demonstrated that basic Assembly Calculus operations (`project` and `reciprocal-project`) form associative memories. The associations of input-output pairs are stored in the weight matrices in the model. Such associative memories are robust in three aspects:

1. The spectral structure of weight matrices shows reliable separation among different stimuli classes.
2. A previously learned assembly can be rapidly reconstructed from a fraction of its neurons.
3. A previously learned assembly can be reconstructed from a corrupted version of the original input.

4 Capacity of Associative Memories in Assembly Calculus

In this paper, we have so far introduced two key operations in Assembly Calculus: 1) `project`, which creates assemblies, and 2) `reciprocal-project` which performs variable binding. We saw in section 3 that the neural circuits that implement these operations are associative memories, and can accommodate stimuli from multiple concept classes. This naturally raises the question of capacity: How many distinct concept classes can we encode in assembly circuits of a given size? How do parameters N, K which control the size of the area and the cap size influence the capacity of these circuits? We aim to answer them in this section. Through empirical measurements, we find that despite the capacity of `project` being higher than that of the Hopfield model, the capacity of `reciprocal-project` is much lower than the capacity of `project`. We attribute the cause of this drop to the fact that `reciprocal-project` involves more than one stage of random projection. In order to alleviate this drop in capacity, we propose a skip connection between the stimulus area and the pointer variable area, as demonstrated in Figure 7B.

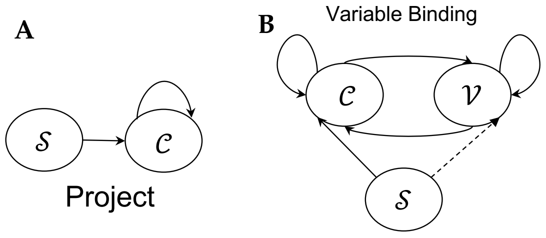

Figure 7: (**A**) Project operation. (**B**) Variable Binding with optional skip connection (dotted arrow).

4.1 Capacity of `Project` Model Exceeds Classical Associative Memory Models

In previous work, as well as in this paper, we have seen that using stimuli from distinct concept classes to train neural circuits for AC operations results in assemblies that are also distinct for each class [14]. We demonstrate an example of assemblies learned for five stimulus classes (using the setup described in section 2.2) through `project` below. In Figure 8B2, the overlap between assemblies of the same class is much larger than the overlap between assemblies of different classes, which means that they can be reliably distinguished. However, we observe that as we add more distinct classes of stimuli during training, the overlap between assemblies of the same classes diminishes while the overlap between assemblies of different classes grows. At a certain critical number of stimuli classes, we see that the assemblies of different classes cannot be distinguished from one another anymore. We refer to this threshold as the *capacity* of the `project` operation. More precisely, an AC operation achieves *capacity* when the average between-class assembly overlap is greater than or equal to the average within-class assembly overlap. Notably, this also corresponds to a linear decoder getting chance accuracy when classifying assemblies of the stored patterns, implying the catastrophic loss of all stored patterns.

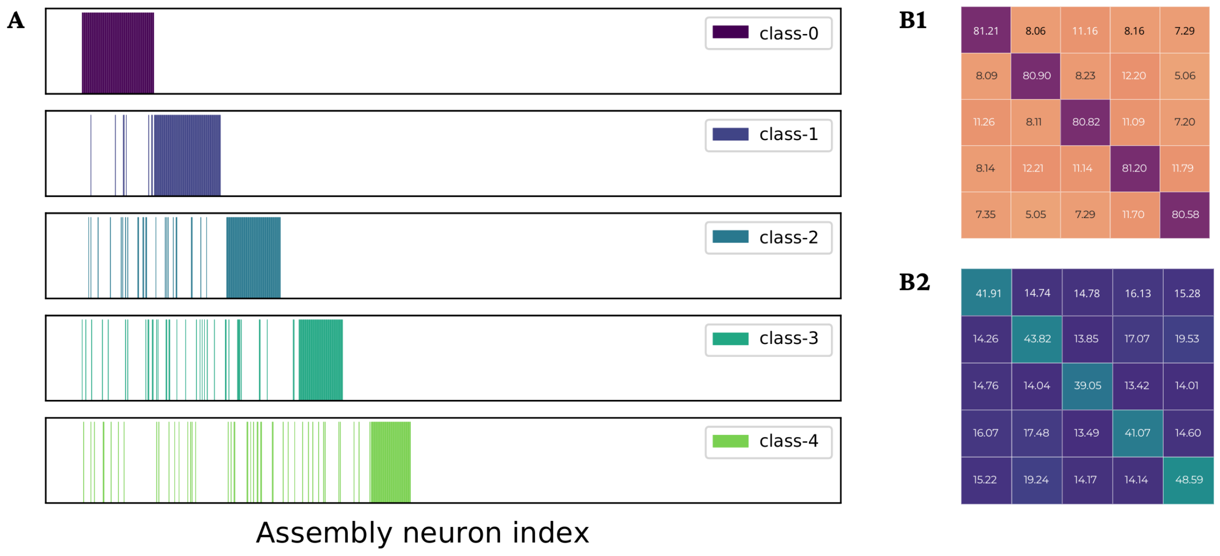

Figure 8: Assemblies learned from five stimulus classes through `project`. In (**A**), we exhibit the distribution of firing probabilities over neurons of the content area $\mathcal{C}$ for each class. In (**B1**) and (**B2**), we show the average overlap (measured in correlation percentage) of different classes of input stimuli (red) and the overlap of the corresponding representations in the assemblies (blue). See the detailed setup and the formation of assemblies under `reciprocal-project` in appendix A.

With this definition of capacity, we explore how the capacity of the `project` operation varies with the number of neurons N and the cap size K. For each setting of the parameters N and K, we train the model with an increasing number of stimuli classes, and track the average within-class similarity of assemblies and compare it to the average between-class similarity of assemblies. We record the capacity (averaged over 5 different trials) as the number of classes when the within-class similarity is less than or equal to the between-class similarity. The exact parameter settings are in appendix E.

As shown in Figure 9A, the capacity of the associative memory created using `project` is $\approx 0.22 \times N$, where N is the number of neurons in each brain area. This is higher than the capacity of Hopfield memories, which was estimated to be $\approx 0.14 \times N$ [23]. The capacity decreases exponentially with K, as shown in Figure 9B.

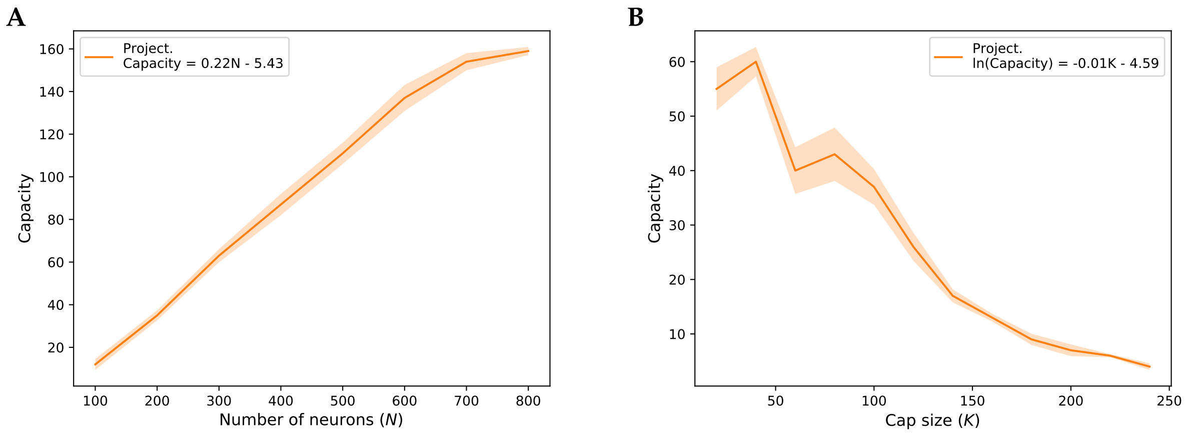

Figure 9: Capacity of `project`, shown by the median number of classes across 5 trials and the corresponding standard error. (**A**) Capacity as a function of N. (**B**) Capacity as a function of cap size K.

4.2 Skip Connections Increase the Capacity of Variable Binding Mechanisms

We now examine the capacity of the `reciprocal-project`, and compare it to the capacity of `project`. Despite the capacity of `project` being competitive with traditional associative memory models, the capacity of the regular `reciprocal-project` model drastically drops to $0.01 \times N$ (Figure 10A, in blue). When measuring how the capacity evolves as a function of the maximum number of active neurons K (Figure 10B), we see that the capacity decreases exponentially with K in all models tested. However, the capacity of the regular `reciprocal-project` model (in blue) is consistently lower than `project` (in orange).

This drop of capacity in `reciprocal-project` can be explained by the fact that the assemblies created in the pointer variable space $\mathcal{V}$ are extremely noisy, due to some loss of information when assemblies are formed in its preceding content space $\mathcal{C}$. See appendix A for the similarity among assemblies in `project` and `reciprocal-project` as a reference. To show that this amplification of noise is inherent to performing two stages of random projections, we compare the capacity of `reciprocal-project` to the capacity of an operation that performs `project` twice - once from the stimulus area $\mathcal{S}$ to the content area $\mathcal{C}$, then from the content area $\mathcal{C}$ to the variable area $\mathcal{V}$. As shown in Figure 10A, B (in purple), for `double-project`, the capacity of the content area $\mathcal{C}$ is approximately the same as in `project`. However, with the noise introduced by the additional layer in `double-project`, the capacity of the variable area $\mathcal{V}$ drops significantly, and becomes similar to the capacity of $\mathcal{V}$ in `reciprocal-project`.

To alleviate the propagation of noise, we propose the addition of a *skip connection* between the stimulus area $\mathcal{S}$ and the pointer variable area $\mathcal{V}$. The addition of the skip connection between $\mathcal{S}$ and $\mathcal{V}$ boosts the capacity of the Variable Binding model by allowing the pointer variable to be directly accessed by the sensory input. This results in an increased capacity $\approx 0.1 \times N$ (Figure 10A, in red). We also measure how the capacity evolves as a function of the number of active neurons K. In Figure 10B, although capacity decreases exponentially with K in all cases, the Variable Binding model with Skip Connections (in red) starts with a larger capacity than the original Variable Binding model (in blue) and is comparable to the `project` model (in orange).

Notably, the introduction of skip connections in our Assembly Calculus model not only addresses the observed capacity drop but also renders the model more *biologically plausible*. A recent study of the connectome of an insect brain [24] has revealed analogous architectural features with connections that skip layers. This finding provides compelling evidence for the biological relevance of skip connections, suggesting that they may serve to increase the brain's computational capacity while overcoming physiological constraints on the number of neurons that limit network depth. We expand the discussion of biological plausibility at the end of the paper.

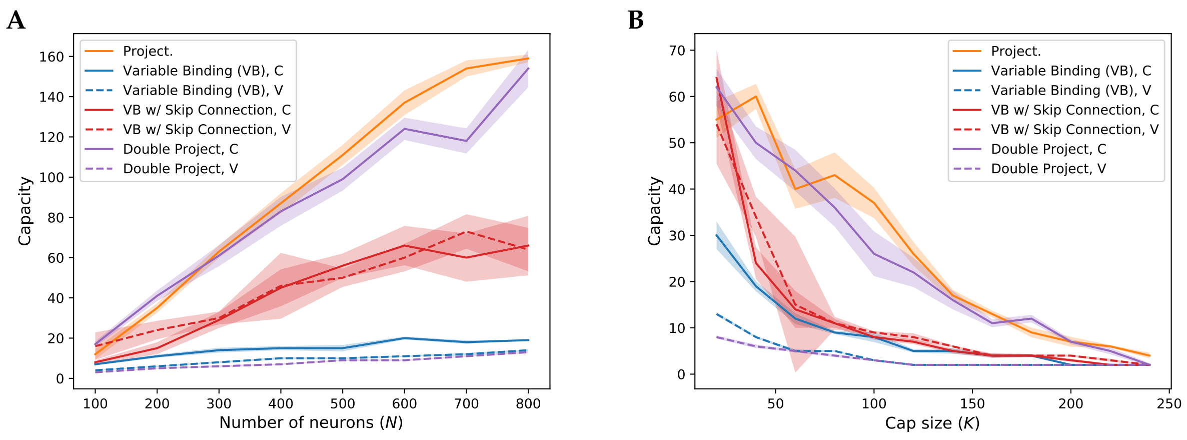

Figure 10: Capacity of different models, shown by the median number of classes across 5 trials and the corresponding standard error. **(A)** Capacity as a function of N for the Variable Binding model without (in blue) and with skip connection (in red). The latter yields a larger capacity. We also show the baseline capacity of `project` model (in orange) and `double-project` model (in purple). **(B)** Capacity as a function of cap size K.

5 Related Work

In this section, we highlight and discuss how our work connects to prior works.

Assembly Calculus: The core framework of our study, Assembly Calculus, is first formally proposed in [1]. It entails the mathematical and theoretical grounding of a repertoire of operations based on assemblies of neurons. The operations include `project`, `reciprocal-project`, `associate`, and `merge`, consistent with recent experimental results [25,26]. Since its introduction, numerous studies have demonstrated the remarkable computational power of its Turing completeness [27], achieved through the use of simple, biologically plausible Hebbian plasticity as opposed to backpropagation. This framework has further opened up new avenues for exploring higher human cognitive functions, including reasoning, planning, and language comprehension. This ranges from classical machine learning tasks such as classifying concept classes [14] (which provides the basis of our experimental setup in this paper), performing boolean operations and supervised learning tasks [28], to other cognition level applications like planning in the block world [12] and language parsing [11].

Associative Memories and their Capacity: Our establishment of the emergence of associative memories in AC is of great significance, enabling us to understand the intermediate-level brain computation more thoroughly and broadly through its important connection with a wider range of well-studied neural network models of associative memory. These include the classical and influential Hopfield networks (HNs) [19,29] and more recently the modern continuous Hopfield networks (MCHNs) [30], which possess close links with self-attention [31] in machine learning; as well as Willshaw Associative Memories [21,32,33], Plate Holographic Reduced Representations [34], Sparse Distributed Memories (SDMs) [20,35,36], Kohonen Memories [17,37], among many others. Notably, the recently proposed Universal Hopfield Networks [38] serve to generalize the operations of such memory networks into a sequence of three operations: similarity, separation, and projection. This framework may facilitate a more modular understanding of AC's random projection and cap (RP&C), considerably a crucial primitive in neural computation and computational learning more broadly [1,39]. Moreover, addressing the inherent capacity constraints of associative memory models is a significant avenue in the past, including Kanerva's SDM, and for future research. It is also the primary motivation behind our characterization of the capacity of AC with respect to the model size and other parameters. AC, like all other associative memory models, exhibits a memory cliff, beyond which the addition of a single pattern results in the catastrophic loss of all patterns. Inspired by the entorhinal hippocampal memory system in mammalian brains, [40] recently proposed Memory Scaffold with Heteroassociation (MESH) as a way to overcome the issue. In our work, we introduce the more intuitive yet biologically plausible idea of connections that skip layers. This type of architecture is characteristic of successful machine learning models, including deep residual learning [41] and U-Net [42]. In particular, skip connections are used to combat information loss and overly abstracted representation over deeper layers due to diminishing gradients in these machine learning models. Its existence is recently confirmed in the connectome of an insect brain [24].

Variable Binding: In this paper we consider circuits that are used to perform binding between content and variable spaces in neural computation [16]. Variable binding implemented by neural assembly models combines the strengths of pointer-based binding [43, 44] - where the variables (or structural roles) are synaptically linked to content, and anatomical binding [45] - where distinct brain areas represent different structural roles and the brain areas contain information related to the content. The pointer-based and anatomical binding models explain different aspects of the experimental findings of Frankland and Greene [46], which show that the functional magnetic resonance imaging activity in specific human brain regions predicts the roles of words in sentences ("agent" vs. "patient", etc.) and also contains information about the attachment of roles to the content of the words. Later work characterizes this variable binding process in a neural model [16]. An alternative model for variable binding is proposed in the convolutional binding of Plate's Holographic Reduced Representations [34] and Kanerva's SDM [20] where neurons and content are represented by high-dimensional vectors, and binding is executed by precise mathematical operations like convolutions. However, these circuits rely on specific synaptic connectivity patterns, which may or may not exist in the brain. While these previous works present different neural architectures for performing variable binding, they do not measure how the capacity of these circuits scales with the model parameters. We focus on the neural assembly-based variable binding model of Muller et al. [16] and show that while the capacity of these models can be limited, we can boost their capacity by adding skip connections.

6 Discussion & Conclusion

In this paper, we study a few basic operations in assembly calculus and show that the corresponding neural circuits can be understood as associative memories. We do so by analyzing the spectral structure of their synaptic weights, as well as demonstrating their ability to retrieve and reconstruct stored neural assemblies. In addition to establishing the correspondence between assembly operations and associative memories, we also empirically measure their capacity and show that their capacity exceeds that of Hopfield memories, even though the linear scaling remains. Our measurements on the capacity of circuits that implement assembly calculus operations can help design models that can scale to larger datasets and a variety of tasks. This would have implications for transfer learning, as well as continual learning of different tasks. Our results show that one can scale a neural assembly model to more tasks by simply increasing the size of each brain area.

Finally, we also show that the capacity of variable binding mechanisms is much lower than the capacity of simple assembly creation, and that their capacity can be boosted by the addition of skip connections. These results indicate that one can construct variable binding models that scale favorably with the size of brain areas by adding skip connections. Our experiments also highlight how models with multiple layers of assemblies as in `reciprocal-project` or `double-project` can amplify the noise in deeper brain areas. The introduction of skip connections offers a streamlined, yet significant, approach to mitigate this influence. This also provides a pathway to design deeper and larger models based on the assembly calculus.

In addition to the computational advantages of skip connections, their presence in the brain is backed up by research on long-ranged connections in the brain [24]. The canonical organization of neocortical circuits [47] can be characterized by a small-world network topology, which consists of dense local clustering and relatively few long-range connections [48]. This topology is claimed to support both specialized and integrated information processing while being effective on wiring costs and enabling high dynamical complexity. Moreover, research shows that long-ranged connections contribute to the optimization of both global wiring cost as well as total processing steps [49]. Studies using cortical thickness measurements in magnetic resonance images have provided further evidence for the existence of long-range connections in the human brain in both intra- and interhemispheric regions. These studies also highlight the robust small-world properties in the human brain anatomical network [50]. Collectively, these findings underscore the biological plausibility and significance of long-range connections in the underlying architecture of complex brain networks.

References

1. C. H. Papadimitriou, S. S. Vempala, D. Mitropolsky, M. Collins, and W. Maass, "Brain computation by assemblies of neurons," *Proceedings of the National Academy of Sciences*, vol. 117, no. 25, pp. 14464–14472, 2020.
2. R. Axel, "Q & A," *Neuron*, vol. 99, p. 1110–1112, 2018.
3. G. Buzsáki., *The Brain from Inside Out.* Oxford University Press, 2019.
4. D. O. Hebb, *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.
5. K. D. Harris, "Neural signatures of cell assembly organization," *Nature reviews neuroscience*, vol. 6, no. 5, pp. 399–407, 2005.
6. G. Buzsáki, "Neural syntax: cell assemblies, synapsembles, and readers," *Neuron*, vol. 68, no. 3, pp. 362–385, 2010.
7. F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.
8. W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.
9. M. Riesenhuber and T. Poggio, "Hierarchical models of object recognition in cortex," *Nature neuroscience*, vol. 2, no. 11, pp. 1019–1025, 1999.
10. C. H. Papadimitriou, "Random projection in the brain and computation with assemblies of neurons," in *10th Innovations in Theoretical Computer Science Conference*, 2019.
11. D. Mitropolsky, M. J. Collins, and C. H. Papadimitriou, "A biologically plausible parser," *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 1374–1388, 2021.
12. F. d'Amore, D. Mitropolsky, P. Crescenzi, E. Natale, and C. H. Papadimitriou, "Planning with biological neurons and synapses," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, pp. 21–28, 2022.
13. G. Marcus, A. Marblestone, and T. Dean, "The atoms of neural computation," *Science*, vol. 346, no. 6209, pp. 551–552, 2014.
14. M. Dabagia, S. S. Vempala, and C. Papadimitriou, "Assemblies of neurons learn to classify well-separated distributions," in *Proceedings of Thirty Fifth Conference on Learning Theory* (P.-L. Loh and M. Raginsky, eds.), vol. 178 of *Proceedings of Machine Learning Research*, pp. 3685–3717, PMLR, 02–05 Jul 2022.
15. R. Legenstein, C. H. Papadimitriou, S. Vempala, and W. Maass, "Assembly pointers for variable binding in networks of spiking neurons," *arXiv preprint arXiv:1611.03698*, vol. 11, 2016.
16. M. G. Müller, C. H. Papadimitriou, W. Maass, and R. Legenstein, "A model for structured information representation in neural networks of the brain," *eneuro*, vol. 7, no. 3, 2020.
17. T. Kohonen, "Self-organization and associative memory," 1989.
18. J. A. Anderson, "A simple neural network generating an interactive memory," *Mathematical biosciences*, vol. 14, no. 3-4, pp. 197–220, 1972.
19. J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities.," *Proceedings of the national academy of sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.
20. P. Kanerva, "Sparse distributed memory and related models," tech. rep., 1992.

21. D. J. Willshaw, O. P. Buneman, and H. C. Longuet-Higgins, "Non-holographic associative memory," *Nature*, vol. 222, no. 5197, pp. 960–962, 1969.

22. J.-e. K. Miller, I. Ayzenshtat, L. Carrillo-Reid, and R. Yuste, "Visual stimuli recruit intrinsically generated cortical ensembles," *Proceedings of the National Academy of Sciences*, vol. 111, no. 38, pp. E4053–E4061, 2014.

23. V. Folli, M. Leonetti, and G. Ruocco, "On the maximum storage capacity of the hopfield model," *Frontiers in computational neuroscience*, vol. 10, p. 144, 2017.

24. M. Winding, B. D. Pedigo, C. L. Barnes, H. G. Patsolic, Y. Park, T. Kazimiers, A. Fushiki, I. V. Andrade, A. Khandelwal, J. Valdes-Aleman, F. Li, N. Randel, E. Barsotti, A. Correia, R. D. Fetter, V. Hartenstein, C. E. Priebe, J. T. Vogelstein, A. Cardona, and M. Zlatic, "The connectome of an insect brain," *Science*, vol. 379, no. 6636, p. eadd9330, 2023.

25. K. M. Franks, M. J. Russo, D. L. Sosulski, A. A. Mulligan, S. A. Siegelbaum, and R. Axel, "Recurrent circuitry dynamically shapes the activation of piriform cortex," *Neuron*, vol. 72, no. 1, pp. 49–56, 2011.

26. J.-e. K. Miller, I. Ayzenshtat, L. Carrillo-Reid, and R. Yuste, "Visual stimuli recruit intrinsically generated cortical ensembles," *Proceedings of the National Academy of Sciences*, vol. 111, no. 38, pp. E4053–E4061, 2014.

27. C. H. Papadimitriou and S. S. Vempala, "Random Projection in the Brain and Computation with Assemblies of Neurons," in *10th Innovations in Theoretical Computer Science Conference (ITCS 2019)* (A. Blum, ed.), vol. 124 of *Leibniz International Proceedings in Informatics (LIPIcs)*, (Dagstuhl, Germany), pp. 57:1–57:19, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018.

28. A. Rangamani and A. Gandhi, "Supervised learning with brain assemblies," *NeurIPS Beyond Backpropagation Workshop*, 2020.

29. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 81, p. 3088—3092, May 1984.

30. H. Ramsauer, B. Schäfl, J. Lehner, P. Seidl, M. Widrich, L. Gruber, M. Holzleitner, M. Pavlovic, G. K. Sandve, V. Greiff, D. P. Kreil, M. Kopp, G. Klambauer, J. Brandstetter, and S. Hochreiter, "Hopfield networks is all you need," *CoRR*, vol. abs/2008.02217, 2020.

31. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

32. D. Willshaw, G. Hinton, and J. Anderson, "Holography, associative memory, and inductive generalization," *Parallel Models of Associative Memory (updated edition), GE Hinton and JA Anderson (eds.)(Hillsdale, Erlbaum, 1989)*, pp. 103–124, 1985.

33. B. Graham and D. Willshaw, "Improving recall from an associative memory," *Biological Cybernetics*, vol. 72, pp. 337–346, 1995.

34. T. A. Plate, "Holographic reduced representations," *IEEE Transactions on Neural networks*, vol. 6, no. 3, pp. 623–641, 1995.

35. L. A. Jaeckel, "An alternative design for a sparse distributed memory," tech. rep., 1989.

36. J. D. Keeler, "Comparison between kanerva's sdm and hopfield-type neural networks," *Cognitive Science*, vol. 12, no. 3, pp. 299–329, 1988.

37. T. Kohonen, P. Lehtiö, J. Rovamo, J. Hyvärinen, K. Bry, and L. Vainio, "A principle of neural associative memory," *Neuroscience*, vol. 2, no. 6, pp. 1065–1076, 1977.

38. B. Millidge, T. Salvatori, Y. Song, T. Lukasiewicz, and R. Bogacz, “Universal hopfield networks: A general framework for single-shot associative memory models,” 2022.

39. S. Dasgupta, C. F. Stevens, and S. Navlakha, “A neural algorithm for a fundamental computing problem,” *Science*, vol. 358, no. 6364, pp. 793–796, 2017.

40. S. Sharma, S. Chandra, and I. R. Fiete, “Content addressable memory without catastrophic forgetting by heteroassociation with a fixed scaffold,” 2022.

41. K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” 2015.

42. O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” 2015.

43. A. Zylberberg, L. Paz, P. R. Roelfsema, S. Dehaene, and M. Sigman, “A neuronal device for the control of multi-step computations,” *Papers in physics*, vol. 5, no. 2, pp. 1–15, 2013.

44. T. Kriete, D. C. Noelle, J. D. Cohen, and R. C. O’Reilly, “Indirection and symbol-like processing in the prefrontal cortex and basal ganglia,” *Proceedings of the National Academy of Sciences*, vol. 110, no. 41, pp. 16390–16395, 2013.

45. K. J. Hayworth and A. H. Marblestone, “How thalamic relays might orchestrate supervised deep training and symbolic computation in the brain,” *bioRxiv*, p. 304980, 2018.

46. S. M. Frankland and J. D. Greene, “An architecture for encoding sentence meaning in left mid-superior temporal cortex,” *Proceedings of the National Academy of Sciences*, vol. 112, no. 37, pp. 11732–11737, 2015.

47. R. J. Douglas and K. A. Martin, “Neuronal circuits of the neocortex,” *Annu. Rev. Neurosci.*, vol. 27, pp. 419–451, 2004.

48. D. S. Bassett and E. Bullmore, “Small-world brain networks,” *The neuroscientist*, vol. 12, no. 6, pp. 512–523, 2006.

49. M. Kaiser and C. C. Hilgetag, “Nonoptimal component placement, but short processing paths, due to long-distance projections in neural systems,” *PLoS computational biology*, vol. 2, no. 7, p. e95, 2006.

50. Y. He, Z. J. Chen, and A. C. Evans, “Small-world anatomical networks in the human brain revealed by cortical thickness from mri,” *Cerebral cortex*, vol. 17, no. 10, pp. 2407–2419, 2007.

Appendix A Assemblies learned for various concept classes

In Figure 8 and 11, we show the assemblies being learned for five stimulus classes through `project` and `reciprocal-project`. For both, $N = 1000, p = 0.1, \beta = 0.1, r = 0.9, q = 0.01$, with 100 samples per class over 5 rounds when forming the assemblies at the test time. We observe the assemblies being formed by `reciprocal-project` is more noisy, and the patterns learned become significantly more sparse as number of classes increase comparing to the ones learned by `project`.

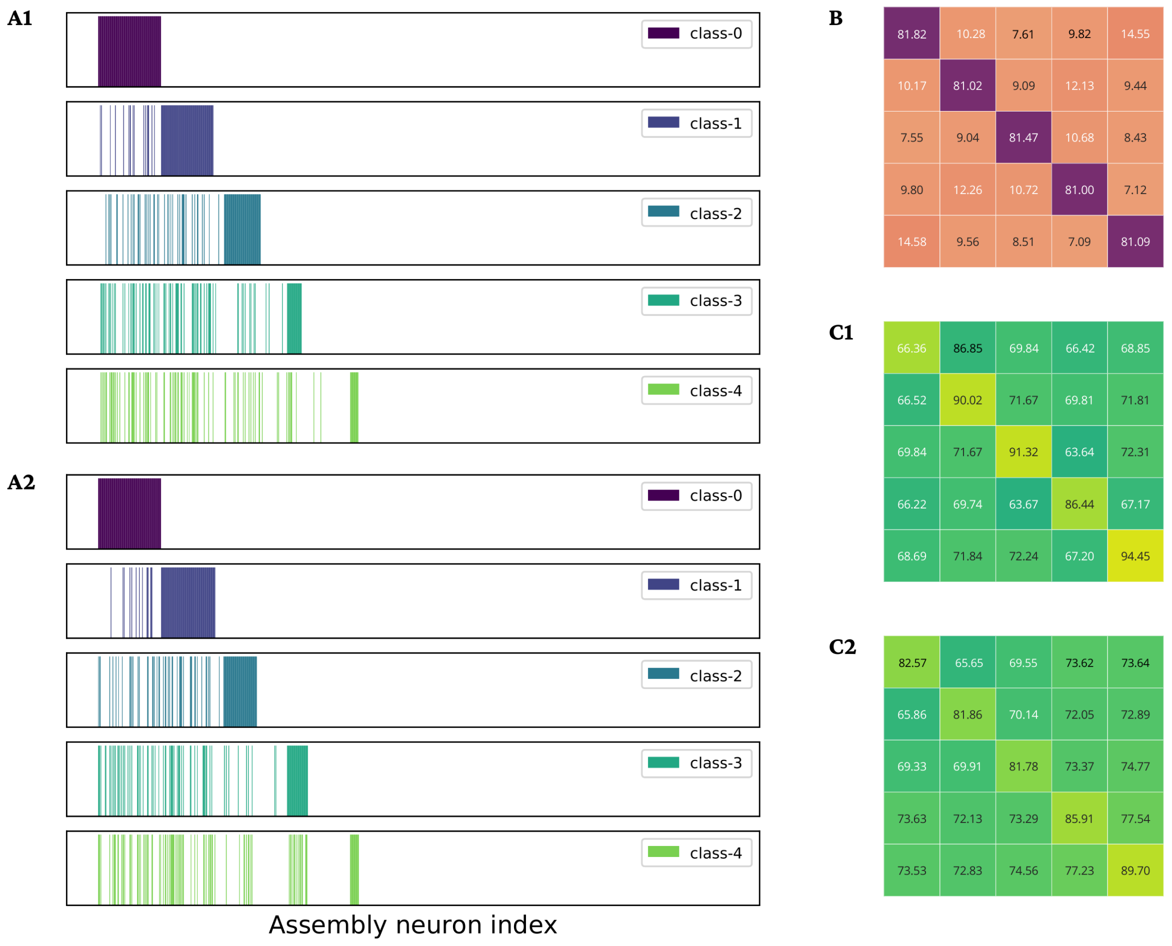

Figure 11: Assemblies learned for five stimulus classes through `reciprocal-project`. In **A1** and **A2**, we exhibit the distribution of firing probabilities over neurons of the content area $\mathcal{C}$ (top) and variable area $\mathcal{V}$ (bottom) for each class. In **B**, we show the average overlap of different input samples (red). In **C1** and **C2**, we show the overlaps of the corresponding representations in the assemblies (green) in content area $\mathcal{C}$ (top) and variable area $\mathcal{V}$ (bottom).

Appendix B The Role of Normalization on Matrix Rank

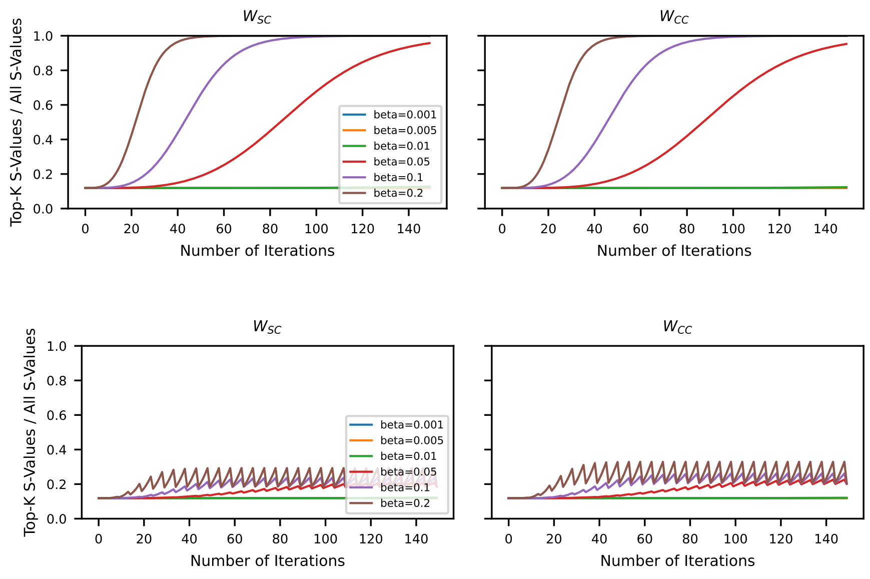

Figure 12: Rank collapsing in `project`: the ratio of the top-K singular value sum over all singular value sum. **Top:** No normalization. **Bottom:** Normalize every 5 iterations.

To track the rank of the weight matrices during `project`, we computed the ratio of the sum of the top-K singular values to the sum of all singular values. A ratio of 1.0 signifies that the rank has collapsed to K, implying a stabilization of the representational potential within the weight matrix. Conversely, a smaller ratio indicates that the matrix retains the capacity for additional information storage within the representational space; however, the acquired information may be less distinguishable from noise compared to a saturated matrix. Hebbian learning propels the matrix towards the system's fixed point, resulting in the rank gradually converging to K as the number of iterations increases; the convergence rate is contingent upon the learning rate β (Figure 12, top).

We noted that weight normalization effectively inhibits the rank from prematurely collapsing to K (Figure 12, bottom). This suggests that, for an equivalent number of iterations, normalization enables the weight matrix to accommodate a greater variety of stimulus classes by preventing its collapse to the fixed point. However, this comes at the expense of potentially weakening the emergent class structure in the normalized weight matrix compared to the unnormalized counterpart. In the following section, we provide evidence that weight normalization can maintain a distinguishable class structure for multiple input classes.

Appendix C Experiment details for pattern completion

Hyperparameters	Specification
α	0.5
N	1000
K	100
p	0.1
β	0.1
m	K
r	0.9
q	0.01
Number of samples per class during training (T_{train})	`np.linspace(4, 14, 10).astype(int)`
Number of assemblies formed per class during testing	50
Number of rounds to form each assembly during testing	5
Number of rounds to *recover* each assembly during testing ($T_{recover}$)	2
Normalization	Normalize after learning each class
Trials	Take median over 5 trials

Table 1: Hyperparameters and other relevant setting used for conducting pattern completion experiment in Section 3.3.

Appendix D Experiment details for assembly recall

Hyperparameters	Specification
N	1000
K	100
p	0.1
β	0.1
m	K
r	0.9
q	0.01
Number of samples per class during training (T_{train})	5
Number of assemblies formed per class during testing	500
Number of rounds to form each assembly during testing	5
Number of rounds to *recover* each assembly during testing ($T_{recover}$)	2
Normalization	Normalize after learning each class
Trials	Take median over 5 trials

Table 2: Hyperparameters and other relevant setting used for conducting assembly recall experiment for perturbing coreset firing rate r or the coreset of each stimulus class in Section 3.3.

Appendix E Experiment details for the capacity measurement

Below we show the set up for hyperparameters and other relevant setting used for the Assembly Calculus model under both `project` or `reciprocal-project` in Section 4. When measuring the model capacity with respect to N (Table 3) or K (Table 4), we vary the hyperparameter of interest (either N or K) while keeping all else constant. Here, N is number of neurons in each brain area of the model, while K is the maximum number of firing neurons per brain area. The synaptic connections between neurons in different areas are drawn independently at random with probability p, while β is the multiplicative Hebbian plasticity learning rate. $\{m, r, q\}$ are specified for the stimulus classes. In particular, m specifies the number of neurons in the coreset of each stimulus class. r is the probability of each neuron in the coreset firing independently, while q is the probability of the neurons outside the coreset firing independently (i.e. noise). During training, the model learns 5 samples drawn independently from a particular stimulus class distribution, updates its parameters, and undergoes a round of normalization, then proceed to learn the next stimulus class. After the parameters are learned, we form 50 assemblies per class at test time; each assembly is formed under 5 rounds of iteration to fully utilize the recurrent connections. We take the median over 5 trials.

Hyperparameters	Specification
$\boldsymbol{N}$	$\mathbf{100 - 800}$
$\boldsymbol{K}$	$\mathbf{50}$
p	0.1
β	0.1
m	K
r	0.9
q	0.01
Number of samples per class during training	5
Number of assemblies formed per class during testing	50
Number of rounds to form each assembly during testing	5
Normalization	Normalize after learning each class
Trials	Take median over 5 trials

Table 3: Hyperparameters and other relevant setting used for training the models and measuring the model capacity with respect to $\boldsymbol{N}$ in Section 4.

Hyperparameters	Specification
$\boldsymbol{K}$	$\mathbf{20 - 240}$
$\boldsymbol{N}$	$\mathbf{250}$
p	0.1
β	0.1
m	50
r	0.9
q	0.01
Number of samples per class during training	5
Number of assemblies formed per class during testing	50
Number of rounds to form each assembly during testing	5
Normalization	Normalize after learning each class
Trials	Take median over 5 trials

Table 4: Hyperparameters and other relevant setting used for training the models and measuring the model capacity with respect to $\boldsymbol{K}$ in Section 4.