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

Yi Xie[*]
Center for Brains, Minds, and Machines
MIT
xieyi@mit.edu

Yichen Li[*]
Department of Psychology
Harvard University
yichenli@fas.harvard.edu

Akshay Rangamani
Center for Brains, Minds, and Machines
MIT
arangam@mit.edu

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 [13] 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. [†]

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

Associative Memory & Hopfield Networks in 2023. NeurIPS 2023 workshop.

1 Introduction

Variable binding is the atomic computation that underlies the ability to separate and assign structural information from content in sensory inputs [8]. Variable binding associates variables to specific values or entities, enabling structured information representation and varied queries like "What color is the block?" and "What object is blue?". In this work, we study the capacity of neural circuits that implement pointer-based variable binding, which separates the content and the pointer into two distinct brain areas. We investigate this architecture in the Assembly Calculus (AC), a computational framework that operates in a level of detail intermediate between the level of spiking neurons and synapses and that of the whole brain models [13]. Assemblies (or Ensembles) of neurons are sets of excitatory neurons that fire together. They are a key data structure in the representation of sensory information in the brain [2], and are believed to encode memories, concepts, words, and other cognitive information [13]. AC is Turing-complete [11] and capable of performing cognitive tasks like language parsing [10] and planning in the block world [3].

In this paper, we establish the two basic AC operations, `project`, and `reciprocal-project`, exhibit *associative memories*. We show that such associative memories yield a clear class structure after receiving multiple classes of stimuli, and can complete patterns based on partial or perturbed inputs. We then 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. Lastly, 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.

2 Assembly Calculus

The model of a brain network in AC consists of several brain areas, each with N binary excitatory neurons. To support inhibition, AC uses a cap-K operation: no more than K out of the N neurons in a specific area can fire at each discrete timestep, where K is the cap size. Afferent connections across brain areas and recurrent connections within each brain areas are initialized independently at random with probability p. Multiplicative Hebbian plasticity increases the strength of the connections between neurons that fire together in consecutive timesteps by a factor of β. We model homeostasis by periodically normalizing the weights of synaptic connections. In this work, we study AC operations that create assemblies (`Project`) and perform variable binding (`Reciprocal-Project`) using stimulus classes as detailed in appendix A.

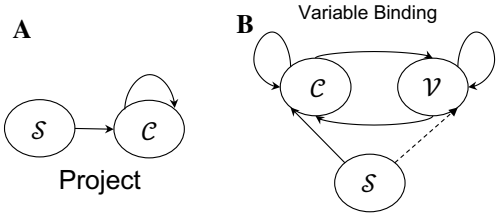

Figure 1: **(A)** Project operation. **(B)** Variable Binding with optional skip connection (dotted arrow).

Project (Figure 1A) `Project` is the primary operation in Assembly Calculus. Through the projection of a stimulus in area $\mathcal{S}$, an assembly forms in a downstream area $\mathcal{C}$ that fires along with the stimulus. The K neurons in area $\mathcal{C}$ with the largest synaptic activation are chosen to fire at each timestep. While the set of neurons that fires varies due to the random afferent and recurrent synaptic connections, Hebbian plasticity helps this process converge to an assembly with high probability [12].

Reciprocal-Project (Figure 1B) `Reciprocal-project` is an operation in Assembly Calculus that performs variable binding. This is achieved through the firing of a stimulus in the input area $\mathcal{S}$, which in turn forms assemblies in the content area $\mathcal{C}$ and in the pointer variable area $\mathcal{V}$. There are forward and backward synaptic connections between $\mathcal{C}$ and $\mathcal{V}$ and recurrent connections within each of the two areas. The bidirectional connections allow the content and variable assemblies to

recall each other. In this paper, we introduce a variation of this operation in which an additional skip connection between the input area $\mathcal{S}$ and the variable pointer area $\mathcal{V}$ is included.

3 Associative Memories Emerge from Assembly Calculus Operations

While previous work [12] has shown that these operations allow convergence to stable sets of assemblies, our objective is to characterize the resulting neural circuits using synaptic weight matrices. We study the patterns and structures that emerge from these neural circuits and characterize them in terms of *associative memories*. We establish the correspondence between associative memories and models in AC in two ways. Firstly, we employ spectral decomposition on synaptic weight matrices and observe clear class structures in the neural circuits. Secondly, we demonstrate that circuits learned during AC operations can reconstruct assemblies from incomplete or noisy stimuli; and, AC can complete patterns, with the firing of a small subset of the assembly being able to activate the whole assembly very accurately, as also observed in [13] with stimuli of a single class. Furthermore, we show that these properties hold when the stimuli come from multiple classes of stimuli.

3.1 Spectral Factors of Synaptic Connections Resemble Associative Memories

In Figure 2, we consider a network that forms content assemblies through `project` from single or multiple classes of stimuli. Different stimulus classes are defined by their distinct coresets of K neurons. As detailed in A, the K neurons within the coreset of a class fire with a high probability (signal), while neurons outside the coreset fire with a small probability (noise). We show that associative memory structure emerges in the synaptic connections learned by Hebbian updates during the operation by implementing Singular Value Decomposition on the afferent and recurrent connectivity matrices. For instance, we examine the decomposition of the afferent connections from $\mathcal{S}$ to $\mathcal{C}$ where $W_{\mathcal{CS}} = U\Sigma V^\top$. We observe that the top right singular vectors (SVs) V_i closely [‡] align with the assemblies c_1, c_2 created by the two stimuli classes in the content area $\mathcal{C}$. Notably, we find that the singular vectors are specifically associated with the inputs and assemblies created by a particular stimulus class. We repeat the analyses on recurrent connectivity weights $W_{\mathcal{CC}}$ and find a similar relationship between singular vectors and assemblies created. The variable binding network exhibits the same behavior, in which the top left SVs U_i are close to the assemblies created in the pointer variable area V. See appendix C for details.

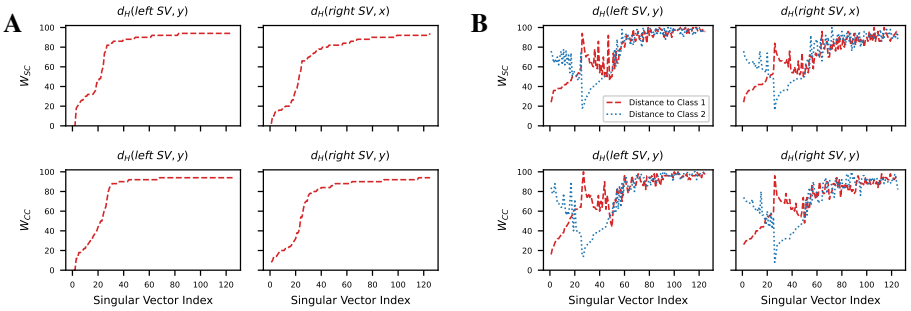

Figure 2: Hamming distance between singular vectors in weight matrices obtained with `project` and the input or output assemblies. During `project`, we normalize the weights of synaptic connections every 5 iteration. The model learns a class until reaches convergence with at most 30 iterations. At test time, we fix the weights and extract assemblies formed after 5 iterations of projection. **(A)** single class of stimuli. **(B)** binary classes of stimuli.

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

In Figure 3, we highlight a key property of associative memories in AC operations: the capacity to recover stored patterns from incomplete assemblies [9] (pattern completion) or perturbed stimuli

[‡]We measure the distance to assemblies c_i as the hamming distance between the assemblies and the K-winner-take-all operation applied to the SVs V_i. This metric is also applied to analyze other weight matrices.

(assembly recall) that the circuit has learned. The ability of AC circuits to reconstruct assemblies from a fraction of their neurons indicates that the recurrent synaptic weights are key to forming associative memories. Similarly, reconstructing assemblies from perturbed stimuli indicates that the afferent synaptic weights also play an important role in forming associative memories. Both demonstrations further support our claim that assembly calculus operations learn associative memories. See more details of the experiment setup in appendix A, E, and F.

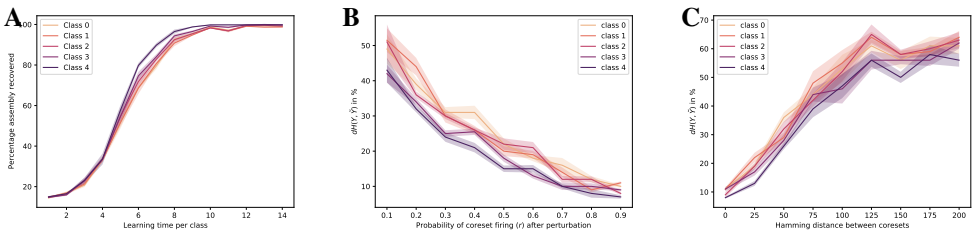

Figure 3: Performance of pattern completion **(A)** and assembly recall (**B** and **C**) for `project`, shown in median values and standard errors across 5 trials. We denote a stack of stimuli x or assemblies y by X and Y. **(A)** Pattern completion across different numbers of training examples for each class. The model forms stable assemblies from few-shot presentations of multiple classes of stimuli. It can recover the original assembly in $T_{\text{recover}} = 2$ iterations when probed with an incomplete assembly with only $\alpha = 50\%$ of its original K winners activated. **(B)** Performance of recalling the original assembly Y (formed under the stimuli X) across perturbations of coreset firing rate r, decreasing from the original $r = 0.9$. At test time, the model receives a stream of corrupted inputs $\tilde{X}$ constructed by varying the coreset firing rate r in each stimulus class within X. The y-axis shows $d_H(Y, \tilde{Y})$ in percentage, the normalized (by K) Hamming distance between the original assemblies Y and of assemblies $\tilde{Y}$. **(C)** Performance of recalling the original assembly Y across manipulations of Hamming distance between the original coreset and the corrupted coreset. The corrupted inputs $\tilde{X}$ are drawn from corrupted coresets $\tilde{S}$ that differ by a certain Hamming distance from the coreset S of the original stimuli X. The y-axis again shows $d_H(Y, \tilde{Y})$ in percentage.

4 Skip Connections Increase the Capacity of Variable Binding Networks

In light of the establishment that Assembly Calculus operations like `project` and `reciprocal-project` essentially learn associative memories, we ask: what is their capacity?

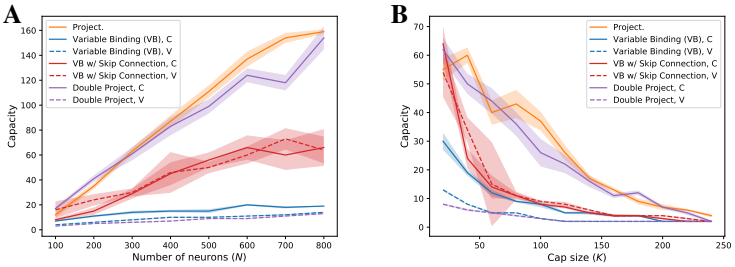

Figure 4: Capacity (number of classes) of different models, shown in median values and standard errors across 5 trials. **(A)** Capacity as a function of N (brain area size) for the Variable Binding model without (in blue) and with a skip connection (in red). The baseline `project` model is shown in light orange. The `double-project` model is shown in purple. **(B)** Capacity as a function of cap size K.

We define the memory *capacity* of these brain networks as the maximal number of distinct stimulus classes whose assemblies can be reliably differentiated from one another. For example, a network for `reciprocal-project` is considered to have reached capacity when any formed assemblies of a class in the content area $\mathcal{C}$ or the pointer area $\mathcal{V}$ become indistinguishable across any other class

of assemblies. This also corresponds to when a linear classifier trained to classify assemblies from different stimulus classes achieves chance-level accuracy.

As shown in Figure 4A, the capacity of the associative memory created using `project` is approximately $0.22 \times N$, where N is the number of neurons in each brain area. This is similar to the capacity of Hopfield memories, which was estimated to be $\approx 0.14 \times N$ [5]. However, the capacity of the regular `reciprocal-project` model (referred to as the VB model for variable binding in Figure 4) dramatically drops to $0.01 \times N$. We postulate that this drop can be explained by the fact that the assemblies created in the pointer variable area $\mathcal{V}$ are extremely noisy, due to some loss of information when assemblies are being formed in its preceding content space $\mathcal{C}$. Appendix B visualizes the similarity among assemblies formed in each downstream brain area for `project` and `reciprocal-project`.

To isolate the cause of capacity drop from the backward connections and simultaneous creations of assemblies in area $\mathcal{C}$ and $\mathcal{V}$ unique to `reciprocal-project` model, we build a `double-project` model. The `double-project` model consists of same brain areas, $\mathcal{S}$, $\mathcal{C}$, and $\mathcal{V}$, but without the backward connections from $\mathcal{V}$ to $\mathcal{C}$. The model instead performs `project` twice in a row sequentially. The effect of noise is still evident in this model. As shown in Figure 4A in purple, while the capacity of the shallower area $\mathcal{C}$ matches that of the `project` model, the capacity of the deeper area $\mathcal{V}$ in `double-project` drops significantly. We infer that the second `project` operation to the deeper area amplifies the noise in the preceding assemblies, thus resulting in a decrease in capacity.

To combat the loss of information, we propose the addition of a skip connection between the input stimulus area $\mathcal{S}$ and the pointer variable area $\mathcal{V}$ to supply the information directly. This results in a tenfold increase of capacity to $\approx 0.1 \times N$, as shown in Figure 4A in red. We also measure capacity's progression as a function of the cap size K (Figure 4B). In all models, capacity decreases exponentially with K. However, the VB model with Skip Connection starts with a larger capacity than the original Variable Binding model, and remains larger across the board. Notably, the capacity of the pointer variable area $\mathcal{V}$ becomes comparable to the shallower area. By allowing the pointer variable to be directly accessed by the sensory input, the skip connection between $\mathcal{S}$ and $\mathcal{V}$ boosts the capacity of the Variable Binding model greatly, while maintaining its variable binding characteristics (e.g., retrieving information in both directions) and being biologically plausible.

The biological relevancy of skip connections is backed up by research on long-ranged connections. A recent connectome study of an insect brain [14] has revealed analogous architectural features with connections that skip layers. Additionally, the canonical organization of neocortical circuits [4] can be characterized by a small-world network topology, which consists of dense local clustering and relatively few long-range connections [1]. This topology is claimed to support both specialized and integrated information processing while being effective on wiring costs and enabling high dynamical complexity. Moreover, [7] shows that long-ranged connections contribute to the optimization of both global wiring cost as well as total processing steps. 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 [6]. Collectively, these findings underscore the biological plausibility and significance of long-range connections in the underlying architecture of complex brain networks.

5 Discussion

We establish that the circuits formed under basic Assembly Calculus operations can be understood as associative memories. We do so by analyzing the spectral structure of the synaptic weights, and by demonstrating their ability to retrieve and reconstruct stored assemblies. We further provide the empirical capacity and show the capacity of assembly creation circuit exceeds that of Hopfield memories, even though the linear scaling remains. Finally, we show that the capacity of variable binding model is much lower than the capacity of assembly creation model, and that their capacity can be boosted by the addition of skip connections. Our 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 that deeper brain areas can amplify the noise. 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 assemblies of neurons.

References

[1] D. S. Bassett and E. Bullmore. Small-world brain networks. *The neuroscientist*, 12(6):512–523, 2006.

[2] G. Buzsáki. Neural syntax: cell assemblies, synapsembles, and readers. *Neuron*, 68(3):362–385, 2010.

[3] 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*, volume 36, pages 21–28, 2022.

[4] R. J. Douglas and K. A. Martin. Neuronal circuits of the neocortex. *Annu. Rev. Neurosci.*, 27:419–451, 2004.

[5] V. Folli, M. Leonetti, and G. Ruocco. On the maximum storage capacity of the hopfield model. *Frontiers in computational neuroscience*, 10:144, 2017.

[6] 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*, 17(10):2407–2419, 2007.

[7] M. Kaiser and C. C. Hilgetag. Nonoptimal component placement, but short processing paths, due to long-distance projections in neural systems. *PLoS computational biology*, 2(7):e95, 2006.

[8] G. Marcus, A. Marblestone, and T. Dean. The atoms of neural computation. *Science*, 346(6209):551–552, 2014.

[9] 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*, 111(38):E4053–E4061, 2014.

[10] D. Mitropolsky, M. J. Collins, and C. H. Papadimitriou. A biologically plausible parser. *Transactions of the Association for Computational Linguistics*, 9:1374–1388, 2021.

[11] C. H. Papadimitriou. Random projection in the brain and computation with assemblies of neurons. In *10th Innovations in Theoretical Computer Science Conference*, 2019.

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

[13] 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*, 117(25):14464–14472, 2020.

[14] 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*, 379(6636):eadd9330, 2023.

Appendix A 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.

Appendix B Assemblies learned for various concept classes

In Figure 5 and 6, 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 that the assemblies formed by `reciprocal-project` are more noisy and the patterns learned become significantly more sparse as the number of classes increases, compared to the ones learned by `project`.

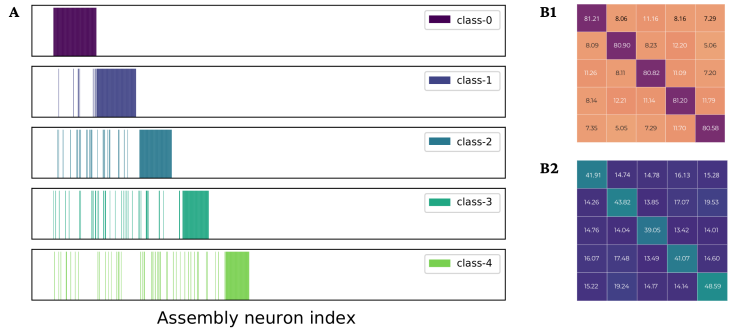

Figure 5: Assemblies learned for five stimulus classes through `project`. In **(A)**, we exhibit the distribution of firing probabilities over neurons of the content area $\mathcal{C}$. for each class In **(B1)** and **(B2)**, we show the average overlap of different input samples (red) and the overlaps of the corresponding representations in the assemblies (blue).

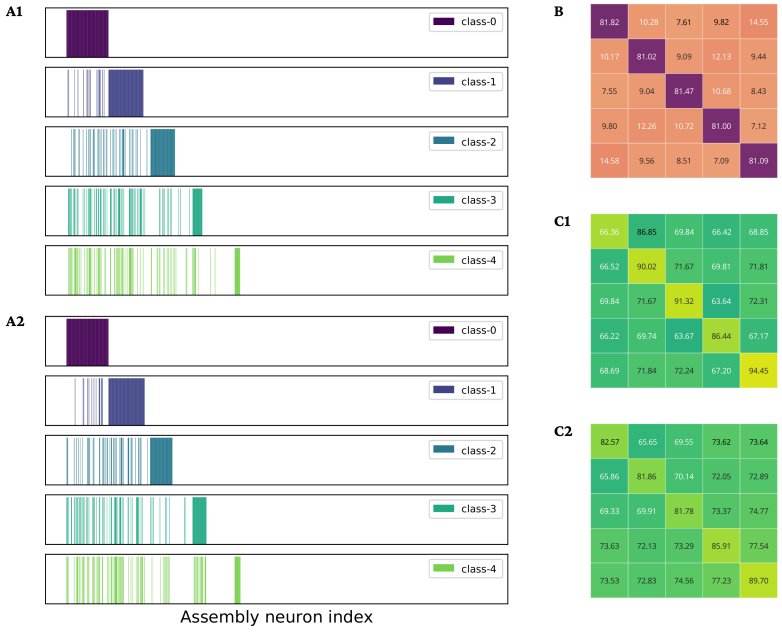

Figure 6: Assemblies learned for five stimulus classes through `reciprocal-project`. In **A1** and **A2**, we exhibit the distribution of firing probabilities over neurons of the content area $\mathcal{C}$ (top) and variable area $\mathcal{V}$ (bottom) for each class. In **B**, we show the average overlap 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 C Further details on spectral factors of synaptic connections

We repeat the SVD Hamming distance visualization for `reciprocal-project`, an operation employing three brain areas to facilitate variable binding. 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 (Figure 7B). 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_{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).

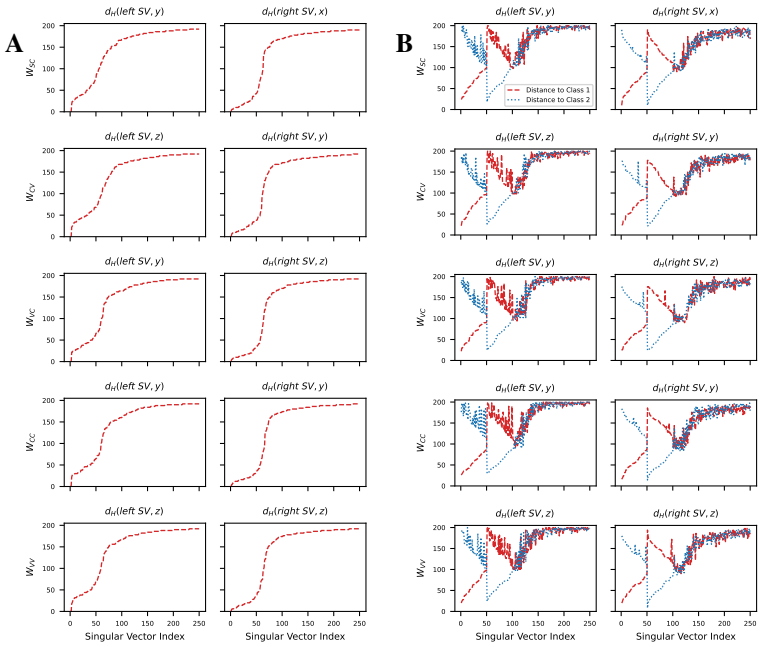

Figure 7: Hamming distance between singular vectors of weight matrices of `reciprocal-project` and the input or output assembly. During learning, we normalize the weights after every 5 iterations. Here, $N = 1000$, $K = 100$, $p = 0.1$, $\beta = 0.1$. The model "learns" a class when reaches convergence with at most 50 iterations. At test time, we extract assemblies formed after 5 iterations. **(A)** A single class of stimuli. **(B)** Two classes of stimuli.

In figures 2 and 7, we observe that the number of singular vectors that are closer than random to the stimuli/assemblies is $\approx K$. This is likely due to the cap-K operation in Assembly Calculus, which limits the rank of the updates to each synaptic weight matrix. However, when examining the multi-class stimuli experiments, we find that the number of singular vectors associated with each stimulus class can be significantly smaller than K. This implies the model can store multiple classes of stimuli-assembly associations, as we explore in section 4. 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 D for more details).

Appendix D The Role of Normalization on Matrix Rank

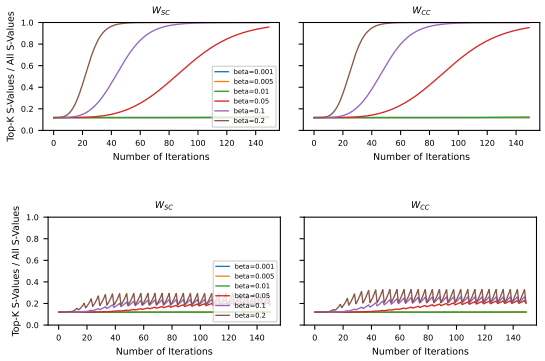

Figure 8: Rank collapsing in `project`: the ratio of the sum of the top-K singular values over the sum of all singular values in a weight matrix. **Top:** No normalization. **Bottom:** Normalize every 5 iterations.

To track the rank of the weight matrices during `project`, we compute 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 than 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 8, top).

We note that weight normalization effectively inhibits the rank from prematurely collapsing to K (Figure 8, 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 sections, we provide evidence that weight normalization can maintain a distinguishable class structure for multiple input classes.

Appendix E Experiment details for pattern completion

Previous work [13] 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, each for T_{train} iterations to form assemblies $\mathbf{y}^m \in \{0,1\}^n, m = 1, \ldots, M$ 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,

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

where S_y denotes the set of K winners in assembly y. The results of this experiment are shown in Figure 3A, 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 Table 1.

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 settings used for pattern completion experiment in Section 3.2.

Appendix F Experiment details for assembly recall

In the patterm completion 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 $y^m \in \{0,1\}^n, m = 1, \ldots, M$ 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 $\tilde{y}^m$.

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 settings used for assembly recall experiments of perturbing coreset firing rate r and the Hamming distance between coresets, in Section 3.2.

Appendix G Experiment details for the capacity measurement

Below we show the setup for hyperparameters and other relevant settings 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 the 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 proceeds 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
$\boldsymbol{N}$	$\boldsymbol{100 - 800}$
$\boldsymbol{K}$	$\boldsymbol{50}$
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 settings used for training the models and measuring the capacity with respect to $\boldsymbol{N}$, in Section 4.

Hyperparameters	Specification
$\boldsymbol{K}$	$\boldsymbol{20 - 240}$
$\boldsymbol{N}$	$\boldsymbol{250}$
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 settings used for training the models and measuring the capacity with respect to $\boldsymbol{K}$, in Section 4.