Learning and Inference in a Lattice Model of Multicomponent Condensates

Cameron Chalk
California Institute of Technology, USA

Salvador Buse
California Institute of Technology, USA

Krishna Shrinivas
Northwestern University, USA

Arvind Murugan
The University of Chicago, USA

Erik Winfree
California Institute of Technology, USA

Abstract

Life is chemical intelligence. What is the source of intelligent behavior in molecular systems? Here we illustrate how, in contrast to the common belief that energy use in non-equilibrium reactions is essential, the detailed balance equilibrium properties of multicomponent liquid interactions are sufficient for sophisticated information processing. Our approach derives from the classical Boltzmann machine model for probabilistic neural networks, inheriting key principles such as representing probability distributions via quadratic energy functions, clamping input variables to infer conditional probability distributions, accommodating omnidirectional computation, and learning energy parameters via a wake phase / sleep phase algorithm that performs gradient descent on the relative entropy with respect to the target distribution. While the cubic lattice model of multicomponent liquids is standard, the behaviors exhibited by the trained molecules capture both previously-observed phenomena such as core-shell condensate architectures as well as novel phenomena such as an analog of Hopfield associative memories that perform recall by contact with a patterned surface. Our final example demonstrates equilibrium classification of MNIST digits. Experimental implementation using DNA nanostar liquids is conceptually straightforward.

2012 ACM Subject Classification Hardware → Biology-related information processing; Theory of computation → Probabilistic computation; Applied computing → Systems biology

Keywords and phrases multicomponent liquid, Boltzmann machine, phase separation

Digital Object Identifier 10.4230/LIPIcs.DNA.30.5

Funding *Cameron Chalk*: National Science Foundation grants 2008589 and 2212546
Salvador Buse: Open Philanthropy Project graduate fellowship, National Science Foundation 2008589 and 2212546
Arvind Murugan: National Science Foundation grants 2239801 and 2317138
Erik Winfree: National Science Foundation grants 2008589 and 2212546

Acknowledgements The authors thank Paul Rothemund, Lulu Qian, Yancheng Du, Emre Alca, Aman Bhargava, Andrej Košmrlj, Inhoo Lee, Mohini Misra, and others for valuable discussion.

1 Introduction

A central goal of the theory of computation is to characterize how computational capabilities relate to mechanistic features of a given model – finite or unbounded memory, one compute head or many, local or global connections – as embodied by models such as Turing machines, finite state automata, Boolean circuits, cellular automata, and the like [36]. For biomolecular

computation, such questions can be phrased in terms of the types of molecules and molecular mechanisms used to construct the computing apparatus – enzymes, DNA, self-assembly, strand displacement, irreversible or reversible reactions, and the like. Ultimately, such an understanding will help us to identify where and how information processing and decision making is occurring in biological cells, as well as to guide the design of cell-scale molecular robotic systems. Considerable progress has been made, for example, understanding how programmable chemical reaction networks (CRNs) can compute in well-mixed solutions [7, 4] and how programmable molecular tiles can compute by self-assembly [8, 28, 10]. As computing architectures, these two cases can be considered to exemplify gas-phase interactions (dilute molecules that interact with each other only intermittently) and solid-phase interactions (molecules that, once they make contact with specific neighbors, remain in contact thereafter). Although many models that have been studied don't neatly fit into these two categories, apparently missing are models that highlight liquid-phase interactions (wherein molecules are in near-constant contact with an ever-changing set of neighbors).

The lack of liquid-phase models for biomolecular computation is not merely a conceptual gap: macromolecules inside cells are tightly packed and in near-constant contact with their ever-changing neighbors, and there is increasing evidence that liquid-liquid phase separation and biomolecular condensates play important roles in biological function [22]. Possible advantages of liquid-phase computation include density (compared to biochemical circuits in dilute solutions), speed (avoiding time waiting for diffusion to bring molecules together), geometry (allowing spatial arrangements to store information similar to solid-phase models), and energy (if the system behavior is governed by equilibrium rather than by consumption of fuels).

To examine such issues in a concrete and tractable model, here we introduce the *Boltzmann liquid* model, which consists of a standard equilibrium lattice model for multicomponent liquids coupled with a learning algorithm derived from classical Boltzmann machine neural network theory that adjusts energy terms to match a target distribution. Our foundational reference point is the classical Boltzmann machine model [1], a constant-temperature generalization of the Hopfield model [15] capable of training hidden units so as to approximate arbitrary probability distributions on binary strings. It has already been observed that detailed balanced chemical reaction networks, in the stochastic limit for well-mixed solutions, can emulate and even generalize Boltzmann machines, and that the learning rule for neural network weights translates into a learning rule for chemical species energies [29, 30]. However, these CRN constructions required $O(N^2)$ species, or worse, to emulate a Boltzmann machine with N neurons, precluding demonstration of powerful computing examples. At the other end of the spectrum, from gas-like to solid-like models, crystalline self-assembly was demonstrated theoretically [27, 48] and experimentally [11] to be capable of Hopfield-like associative recall and pattern recognition due to an analogous quadratic energy function. A hint at the extra power available in liquid-phase systems is that with N species, there are already a full $O(N^2)$ interaction energies – which potentially could be put in correspondence with the $O(N^2)$ synaptic weights in a neural network. Another hint comes from Bruck's proof of the convergence of Hopfield networks [5], which envisions neurons being physically separated into an "ON" group and an "OFF" group such that the system energy is evaluated at the interface (a min cut), analogous to phase separation. Further parallels to neural networks come from the observation that multicomponent liquids are capable of supporting multiple distinct phases – combinations of different species that separate from each other in a manner somewhat analogous to Hopfield's associative memories [39, 40, 45]. Insights at the abstract level have promise for immediate impact on experimental investigations with DNA nanotechnology, due to the robust and programmable DNA nanostar motif [3, 19, 35], and may be helpful advancing the vision of cell-scale molecular robotics and smart droplets [46, 44, 26].

In Section 2, we detail our model and equilibrium sampling methods. Section 3 introduces Hopfield networks and Boltzmann machines in order to introduce key concepts of learning and information processing by equilibrium systems. Section 4 outlines our perspective on learning and inference in the Boltzmann liquid model. Sections 5, 6, 7, and 8 showcase varied applications of our learning rule, along with modes of inference relevant to molecular environments. This initial variety of results suggests a rich information processing capability inherent in the equilibrium distributions of biomolecular condensates, and solidifies our proposed learning rule as a general method for exploring the range of possible phenomena within this design space.

2 Model and equilibrium sampling methods

The Boltzmann liquid model considers a finite discrete set of positions, V, which in this paper we will take to be the three-dimensional lattice $V = [1, L]^3$. If positions $p_1 \in V$ and $p_2 \in V$ are neighbors, meaning a molecule at p_1 will be in contact with a molecule at p_2, we say $p_1 \sim p_2$. For our cubic lattice, that means the positions are face-adjacent, i.e. each cube has six von Neumann neighbors (except at the boundaries, since we do not use periodic boundary conditions). A lattice configuration σ has one molecule from a set M of molecule types assigned to each position: $\sigma : V \to M$. The set M generally includes the solvent and the solvent is treated as any other molecule type. Thus, the model assumes equal-sized molecules.

For equilibrium properties, the essential property is the energy for a given configuration, $G(\sigma)$. Following Jacobs and Frenkel [17, 18], we use a standard lattice-gas generalization of the Ising model [21] for multicomponent liquids:

$$G(\sigma) = \frac{1}{2} \sum_{p_1 \in V} \sum_{\substack{p_2 \in V \\ \text{s.t. } p_1 \sim p_2}} G_{\sigma(p_1),\sigma(p_2)} + \sum_{p \in V} G_{\sigma(p)} \ , \tag{1}$$

where $G_{i,j} = G_{j,i}$ is the energy for a molecule of species $i \in M$ interacting with a neighboring molecule of species $j \in M$, and where G_i is the internal energy (or chemical potential) of species i. We consider $G_{i,j}$ and G_i to be the parameters of the Boltzmann liquid model that can be tuned arbitrarily to obtain a range of behaviors. The convention is that more negative $G_{i,j}$ and G_i are more favorable.

When working with this energy function, it will often be convenient to express it using a notation that makes the dependence on the model parameters more salient. To do so, we will need the Kronecker delta function, which returns one if its subscripts are equal, else zero:

$$\delta_{i,j} = \begin{cases} 1 & i = j \\ 0 & i \neq j \ . \end{cases}$$

A configuration σ induces a value $n_{i,j}$ for all $i, j \in M$, which is the number of neighbor interactions between molecules i and j in the configuration. We define:

$$\hat{n}_{i,j} = \sum_{p_1 \in V} \sum_{\substack{p_2 \in V \\ \text{s.t. } p_1 \sim p_2}} \delta_{\sigma(p_1),i} \delta_{\sigma(p_2),j} \ .$$

As written, $\hat{n}_{i,j}$ double counts the case $i = j$, which we correct as:

$$n_{i,j} = \begin{cases} \frac{1}{2}\hat{n}_{i,j} & i = j \\ \hat{n}_{i,j} & i \neq j \ . \end{cases}$$

Additionally, a configuration σ induces a count of each molecule type in the lattice,

$$n_i = \sum_{p \in V} \delta_{\sigma(p),i} \ .$$

Finally, we can see that lattice configuration σ has the corresponding energy:

$$G(\sigma) = \sum_{i \leq j \in M} n_{i,j} G_{i,j} + \sum_{i \in M} n_i G_i \tag{2}$$

wherein each independent energy parameter appears exactly once.

Assuming a dynamics that obeys detailed balance with respect to this energy, the Boltzmann distribution provides an analytical formula for the probability of a given configuration at equilibrium in terms of its energy. Key to this statistical mechanical approach is knowing the set of states Ω that are reachable from the initial state, i.e., the statistical ensemble that will be explored by the dynamics. The two ensembles we consider in this work are the canonical (a system which exchanges only heat with a reservoir, and keeps its molecule counts fixed) and the grand canonical (a system which exchanges both heat and molecules with a reservoir). In the grand canonical ensemble, the set of configurations is all possible configurations, $\Omega = \mathcal{G} = \{\sigma \mid \sigma : V \to M\}$. In the canonical ensemble, molecule type counts are conserved, so that the set of configurations given a vector $\vec{c}$ of counts of each molecule type is given by $\Omega = \mathcal{C}_{\vec{c}} = \{\sigma : V \to M \ \mid \forall i \in M, n_i(\sigma) = \vec{c}(i)\}$, where $n_i(\sigma)$ is the count of molecule i in σ. It is important to note, then, that the per-molecule energy contribution $n_i G_i$ of Equation 2 is the same for all configurations in the canonical ensemble, and thus we fix G_i to zero for canonical simulations. Given either of the ensemble assumptions, the probability of a configuration at equilibrium is given by:

$$P(\sigma) = \frac{1}{Z} e^{-G(\sigma)/kT}, \quad \text{where } Z \text{ is the partition function,} \quad Z = \sum_{\sigma \in \Omega} e^{-G(\sigma)/kT}$$

and kT is the reference energy at the chosen temperature.

Simulations take place from an initial configuration by accepting proposed moves according to the Metropolis-Hastings criteria: a move and its corresponding reverse move must be proposed with the same probability, and a move which changes energy by ΔG is accepted with probability $\min(1, e^{-\Delta G/kT})$. A simulation using this criteria satisfies detailed balance, and thus in the limit samples from the Boltzmann distribution. The type of moves proposed in the simulations depends on whether the ensemble is canonical or grand canonical. We emphasize that in this work, our aim is to sample the equilibrium distribution, not necessarily the exact kinetics or dynamics. In the canonical dynamics, proposals are swaps between any two positions in the lattice. We found empirically that such long-range swaps, while resulting in different kinetics than more physical local swaps of neighboring or near-neighboring sites, allow more efficient equilibrium sampling. In the grand canonical dynamics, proposals are to swap a molecule in a position with another molecule type. Other choices could work as well; beyond the canonical vs. grand-canonical distinction, we do not expect the choice of proposals to impact our results, to the extent that our simulations achieve near-equilibrium sampling. Additionally, we propose and accept/reject moves in parallel using GPU-accelerated algorithms, described in detail in Section A.1.

3 Hopfield networks and Boltzmann machines

In this section we describe Hopfield networks and Boltzmann machines, two simplified models of neurons whose discovery and study inspire this work. In particular, two key insights in the study of Boltzmann machines underlie our perspective on multicomponent condensation. The first key is a learning rule which can learn parameters for all interactions even when a target distribution is specified by only a subset of free neurons, i.e., it can learn "hidden unit" behaviors within a fully-connected network that can perform omnidirectional computation without a feed-forward input-to-output direction being architecturally-defined. Second, that *clamping*, or fixing a subset of neurons to a target state, results in the remainder of the system exploring a conditional probability distribution, i.e., performing inference from any subset of variables to any other subset of variables as its main method of information processing. These two key insights are described further in this section after some preliminaries.

Hopfield [15] showed that a model of N simplified, interconnected neuron-like elements (a Hopfield network) could perform associative recall, where a full state could be recovered from a partial or noisy state. Hopfield assigned the following energy to his system to describe the stability of states:

$$E(s) = -\frac{1}{2}\sum_{i,j} w_{i,j} s_i s_j - \sum_i b_i s_i, \tag{3}$$

where $s \in \{-1, 1\}^N$ indicates which of the N neurons are "ON", $w_{i,j}$ denotes a real-valued *weight* between each neuron, and b_i is a bias applied to each neuron. Here, $w_{i,j} > 0$ encourages neurons i and j to have the same state, while $b_i > 0$ encourages neuron i to be "ON".

To "learn" or "memorize" a set of states S, called memories, Hopfield proposed a simple Hebbian fire-together, wire-together rule: $w_{i,j} = \frac{1}{|S|}\sum_{s\in S} s_i s_j$. Thus, each weight is strengthened when neurons are correlated in the memory, and weakened when they are anticorrelated. Given a dynamics that walks downhill in the energy landscape, starting at a state close to a memory (e.g., a memory with some noise applied, or a partial memory), the dynamics recalls the full memory. This occurs because the Hebbian learning creates wells in the energy landscape corresponding to the memories. With the above Hebbian learning rule, attempts to store too many memories (more than a capacity [2] that is linear in N) results in an energy landscape dominated by spurious wells and thus failure to correctly recall memories. This capacity can be improved by "unlearning" with anti-Hebbian adjustments when spurious memories are encountered [16, 38].

Towards allowing the system to explore more complex probability distributions, Hinton and Sejnowski [14] proposed using the same architecture and energy function with a stochastic dynamics to escape local minima. With states $s \in \{0, 1\}^N$ and energy function as in Equation 3, neurons are selected at random and set to 1 with probability $P(s_i = 1) = \frac{1}{1+e^{\Delta E_i/T}}$ and set to 0 otherwise, where ΔE_i is the change in energy caused by changing neuron i to 1 if it was 0 given the current state of the rest of the neurons, and T is a parameter analogous to temperature. From this dynamics, the steady-state probability for the Boltzmann machine is given by the Boltzmann distribution:

$$P(s) = \frac{1}{Z}e^{-E(s)/T}, \quad \text{where } Z \text{ is the partition function,} \quad Z = \sum_{s\in\{0,1\}^N} e^{-E(s)/T}.$$

Thus, the Boltzmann machine imitates the physical phenomenon of an equilibrium system stochastically exploring its Boltzmann distribution.

Looking beyond associative recall, Ackley, Hinton, and Sejnowski [1] proposed a learning rule for parameters of the Boltzmann machine to push the Boltzmann machine's distribution over states, P, towards a target distribution Q. A key insight of the authors was presenting

the learning rule in terms of marginal distributions over a subset of neurons, called *visible* neurons, and showing that the remaining neurons, called *hidden* neurons, would have their parameters updated according to the learning rule to aid the visible units in representing their target marginal distribution. In fact, any distribution over a set of visible neurons can be learned successfully given enough hidden units [43, 47]. The classical Boltzmann machine laid the groundwork for modern restricted Boltzmann machines (RBM) and deep Boltzmann machines (DBM) [33] that impose restricted input/output information flows that enable faster and more effective learning algorithms, but for our purposes the classical learning rule is more relevant because it allows us to generalize to the unrestricted energy model of our Boltzmann liquid.

The classical Boltzmann machine learning rule is as follows. Neurons are divided into visible neurons v and hidden neurons h, so the state can be written by concatenation as $s = vh$. Given a target distribution Q_v over the visible neurons, we aim to find parameters for all $w_{i,j}$ and b_i (including parameters involving hidden neurons) such that the marginal distribution over visible neurons, P_v, is equal to Q_v. It is helpful to define the following distribution, $Q(vh) = \sum_v Q_v(v)P(h|v)$. This describes the Boltzmann machine with hidden units running freely and the visible units clamped to take on the target distribution Q_v; the marginal distribution of Q for visible variables is Q_v. The learning rule is described as a gradient descent which minimizes the relative entropy, also known as the Kullback-Leibler divergence:

$$D_{\mathrm{KL}}(Q_v \,||\, P_v) = \sum_{v \in \{0,1\}^{|v|}} Q_v(v) \log \frac{Q_v(v)}{P_v(v)} \ .$$

The Kullback-Leibler divergence, while not a metric in the strict mathematical sense, is always nonnegative and is zero when $P_v = Q_v$. As derived in [1], the gradient is:

$$\frac{\partial D_{\mathrm{KL}}(Q_v \,||\, P_v)}{\partial w_{i,j}} = \langle s_i s_j \rangle_P - \langle s_i s_j \rangle_Q \ ,$$

where $\langle s_i s_j \rangle$ is the average correlation between neurons i and j in the respective distribution. Remarkably, even though it is only the relative entropy of the marginal distributions that is being minimized, the averages here are with respect to the full distributions over both visible and hidden units. Thus, the averages are well-defined even when i and/or j are hidden, so hidden units can be trained. Since we want to minimize the divergence, this suggests the gradient descent:

$$\frac{\delta w_{i,j}}{\delta t} = \langle s_i s_j \rangle_Q - \langle s_i s_j \rangle_P \ .$$

An analogous rule for adjusting b_i follows immediately.

The second key insight is the *clamping* of neurons, wherein visible neurons are clamped to a target configuration or distribution. The result of clamping the visible neurons to a configuration v^* is that the Boltzmann machine then takes on the conditional probability distribution $P(h \mid v = v^*)$. So, upon clamping some neurons, the remaining neurons compute the conditional probability (inference) by relaxing to equilibrium, highlighting a connection between inference and energy minimization. Furthermore, the clamped neurons needn't be exactly the visible units that were used for training; clamping any subset of units (visible or hidden) results in inference of the corresponding conditional distribution on the unclamped units. This is what we mean by omnidirectional computation.

To illustrate inference and the learning rule, consider the MNIST database of handwritten digits, a standard machine learning benchmark. In this database, there are 10 classes of digits, zero through nine, and each digit image is represented by a 28×28 array of real

values in $[0, 1]$. To be represented by a Boltzmann machine, the images are mapped to binary arrays (e.g., by clipping values above or below 0.5 to 1 and 0, respectively). Each pixel is then assigned a neuron. Commonly, ten neurons are assigned to the classes, in a "one-hot" representation where the correct class for the digit is set to one, and the others set to zero. Any number of additional neurons can be included as hidden neurons.

To train, the image and class neurons are clamped to samples from the MNIST training set. Thus, the target distribution Q_v over visible neurons is the uniform distribution of MNIST training set samples mapped to visible neurons. Practical implementations of the learning rule sample $\langle s_i s_j \rangle_Q$ and $\langle s_i s_j \rangle_P$ and use Euler integration with a learning rate ϵ to approximate the gradient descent. Sampling $\langle s_i s_j \rangle_Q$ can be achieved by Markov-chain Monte Carlo (MCMC) sampling with the Boltzmann machine stochastic update rule, with the visible units clamped to MNIST training samples. Then, sampling $\langle s_i s_j \rangle_P$ is done by leaving the visible neurons unclamped, or *free*, and using (typically) the same sampling method. This classical Boltzmann machine learning algorithm can be interpreted as alternating "wake phase" Hebbian learning and "sleep phase" anti-Hebbian unlearning: while the system is awake and being clamped by the environment, it strengthens its weights based on the correlations $\langle s_i s_j \rangle_Q$. Then, while asleep and "dreaming", with its neurons unclamped, the system weakens or unlearns correlations $\langle s_i s_j \rangle_P$.

Once trained, the Boltzmann machine can sample from P_v, which should be close to Q_v. For the MNIST example, digit recognition occurs by clamping just the image neurons, and letting the class (and hidden) neurons run free. Then (if trained successfully, i.e., with enough hidden neurons, accurate sampling, and small enough learning rate), the free class neurons will tend towards the one-hot representation with the correct digit on and the others off. An equally valid inference would follow from clamping just the class neurons, and sampling the image neurons. If other partial information were available, e.g. that the digit is not 7 and only the upper part of the image is known, then clamping those respective neurons would again lead to sampling the correct conditional probability distribution for the unknown neurons. In other words, no neurons are specified explicitly as input or output, and the machine can interact with the environment generically via clamping.

4 From Boltzmann machines to Boltzmann liquids

In this section we extend insights gleaned from Boltzmann machines to understand learning and inference in our molecular lattice model, and identify and program the modes of information processing available to liquid-phase molecular systems.

In general, we will consider distributions over observables (i.e., distributions described in terms of macrostates). Letting Ω be the set of configurations of the ensemble (canonical or grand canonical), an observable is any function $a : \Omega \to A$. For example, a could map a configuration σ to the count of the third species, $n_3(\sigma)$, or to the identities of species in positions (x_1, y_1, z_1) and (x_2, y_2, z_2). An observable a induces a set of macrostates $M_a(\alpha) : \{\sigma \in \Omega \mid a(\sigma) = \alpha\}$, e.g., the set of all configurations with the count of the third species equal to α. The marginal probability of a given macrostate is the sum of probabilities of corresponding microstates: $P_a(\alpha) = \sum_{\sigma \in M_a(\alpha)} P(\sigma)$.

If the environment clamps the system to an observable outcome $\alpha \in A$, the remainder of the system takes on the conditional probability, computing inference in the same respect as the Boltzmann machine:

$$P_\alpha(\sigma) = P(\sigma \mid \alpha) = \begin{cases} \frac{1}{Z_\alpha} e^{-G(\sigma)} & \sigma \in M_a(\alpha) \\ 0 & \sigma \notin M_a(\alpha) \end{cases}$$

where $Z_\alpha = \sum_{\sigma \in M_a(\alpha)} e^{-G(\sigma)}$.

Given a target distribution of observable outcomes Q_a, the distribution of the lattice clamped to Q_a is given by $Q(\sigma) = \sum_{\alpha \in A} Q_a(\alpha) P_\alpha(\sigma)$. What we prove is the following: letting $\langle n_{i,j} \rangle_D$ be the average value for $n_{i,j}$ in the distribution D,

$$\frac{\partial D_{\mathrm{KL}}(Q_a \,||\, P_a)}{\partial G_{i,j}} = \langle n_{i,j} \rangle_Q - \langle n_{i,j} \rangle_P \ ,$$

which suggests the gradient descent:

$$\frac{\delta G_{i,j}}{\delta t} = \langle n_{i,j} \rangle_P - \langle n_{i,j} \rangle_Q \ .$$

Note that our parameters $G_{i,j}$ are more favorable when more negative, so the learning rule is inverted compared to the Boltzmann machine's rule (but still "strengthened when clamped, weakened when free"). The same proof applies to G_i, suggesting $\frac{\delta G_i}{\delta t} = \langle n_i \rangle_P - \langle n_i \rangle_Q$. The derivation is in Appendix A.3.

In practice we approximate $\langle n_{i,j} \rangle$ for distributions Q or P via Markov-chain Monte Carlo (MCMC) and use Euler integration with a learning rate ϵ to approximate the gradient descent. For example, when clamping voxel positions of the lattice to particular molecule types, in a "wake" phase the visible positions are clamped according to Q while the free positions equilibrate, and samples are accumulated for $\langle n_{i,j} \rangle_Q$ and $\langle n_i \rangle_Q$, thus leading to Hebbian strengthening of interaction energies; and in a "sleep" phase no positions are clamped and all units equilibrate to accumulate samples $\langle n_{i,j} \rangle_P$ and $\langle n_i \rangle_P$, thus leading to anti-Hebbian weakening of interaction energies.

In the following sections, we provide examples of the learning rule, inference via clamping, and generalization beyond the target distribution (a common phenomenon in machine learning: for example, trained on MNIST, a Boltzmann machine can classify digits from a holdout set it was not trained on). We commonly use the terminology visible/hidden and clamped/free from Boltzmann machine literature, to refer to molecule types or positions that are part of the observable (visible, clamped) or not part of the observable (hidden, free).

5 Learning and inference of a structural observable

Here we show our learning rule applied to a distribution specifying molecular structure. This is motivated by recent observations that many biological condensates, such as the nucleoli [12] and paraspeckles [13], exhibit organization into compositionally distinct internal layers. The "avocado" example, inspired by the 3-layer architecture of the nucleolus [12] and prior work on designing liquid-liquid phase separation [24, 23], highlights that the learning rule can be used to find intermolecular energies which yield distributions with complex structural properties, like multiple shells forming around a core (Figure 1).

This simulation takes place in the canonical ensemble, with a fixed number of molecule counts. The wake phase observable makes all species visible and clamps to an avocado shape: Q is a point distribution over configurations, i.e., a single configuration of a perfectly spherical avocado centered in the lattice. Thus, no simulation needs to occur to collect $\langle n_{i,j} \rangle_Q$. To collect $\langle n_{i,j} \rangle_P$ during the sleep phase, the avocado is shuffled (i.e., a uniform random permutation of the lattice with counts equal to the ideal avocado is selected) as the starting state for MCMC, and samples are collected after enough steps to approximately equilibrate the system. See Figure 7 for an example plot of configuration energy per simulation step.

The avocado example provides an intuitive introduction to a central concern of our work: that the learned distribution P is unlikely to match the target distribution Q, because the Boltzmann liquid model is not capable of representing arbitrary distributions. For example,

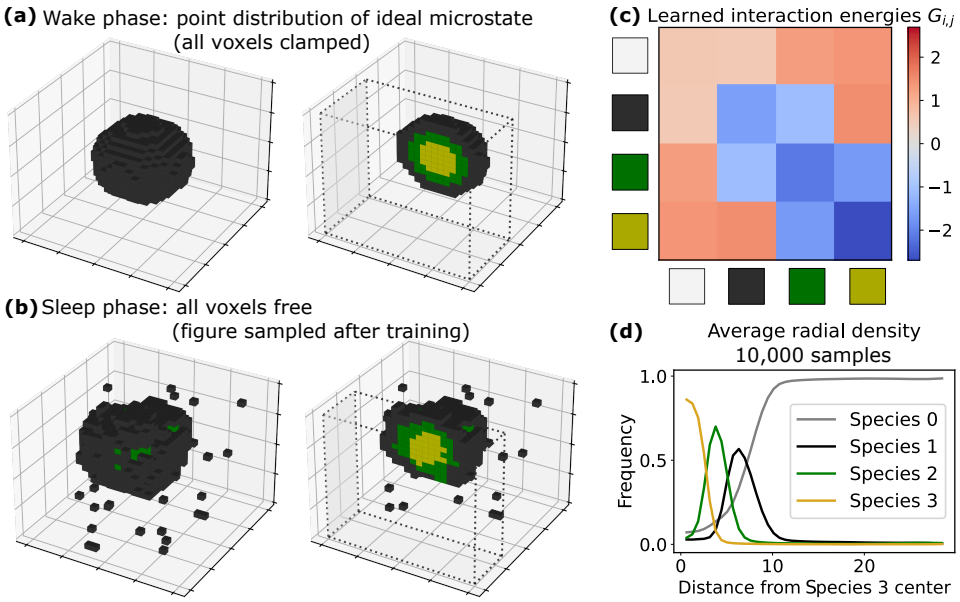

Figure 1 Learning an "avocado" structure with an outer, middle, and inner layer in a canonical ensemble. **(a)** The target distribution Q is a point distribution for the target configuration shown at the top ($24 \times 24 \times 24$ lattice). Dotted boundaries indicate the voxels within are not drawn, to expose internal structures. **(b)** A sample selected from a canonical simulation after training. **(c)** Interaction energies after 80 steps of learning. **(d)** Average distances from the center of mass of Species 3 for each species were collected over 100 independent MCMC trials, each with 100 samples.

although the target distribution Q consists of a point distribution of a single perfectly-centered avocado, any set of energies will give the same probability for states in which the avocado is shifted to the left or right (without touching the walls), and will give similar energies for slightly distorted avocados. So P will not be a point distribution. In machine learning, the limited power of a model often implies that after learning, the machine must generalize from the provided data and assign probabilities to states that were not visited during training. Here, the generalization implied by the learned P includes many "avocado-like" states, for example. Key is the fact that the learned interaction energies yield distributions with desired statistics despite the impossible target, as shown in Figure 1c, which validates the statistical integrity of the layered structure across many MCMC samples: the radial densities suggest that the inner, middle, and outer layer structure is a general feature of the equilibrium.

The learned interaction energies shed light on physical features required to design avocado-like structures beyond requiring strong self-interactions that drive condensation ($G_{i,i} < 0$). The nested morphology we learn requires correct ordering of strengths of inter-species interactions, and thus surface tensions between phases [24, 23].

Figure 2 shows examples of inference achieved by the same interaction energies trained on the ideal point distribution. These examples explore how the model generalizes: clamping a set c of voxels (which may be the visible or hidden positions from training, or another subset) will induce the conditional probability $P(\sigma \mid c)$, which may result in high (conditional) probability for states that are very low probability in the unclamped (free) distribution P. Figure 2a shows that the spatial distribution of the avocado can be localized via a surface clamp. Figure 2b emulates that a patterned polymer is enough to centralize the spatial distribution of the condensate, suggesting exploration into the role of polymers in condensate formation. Figure 2c highlights the role of inference as conditional probabilities

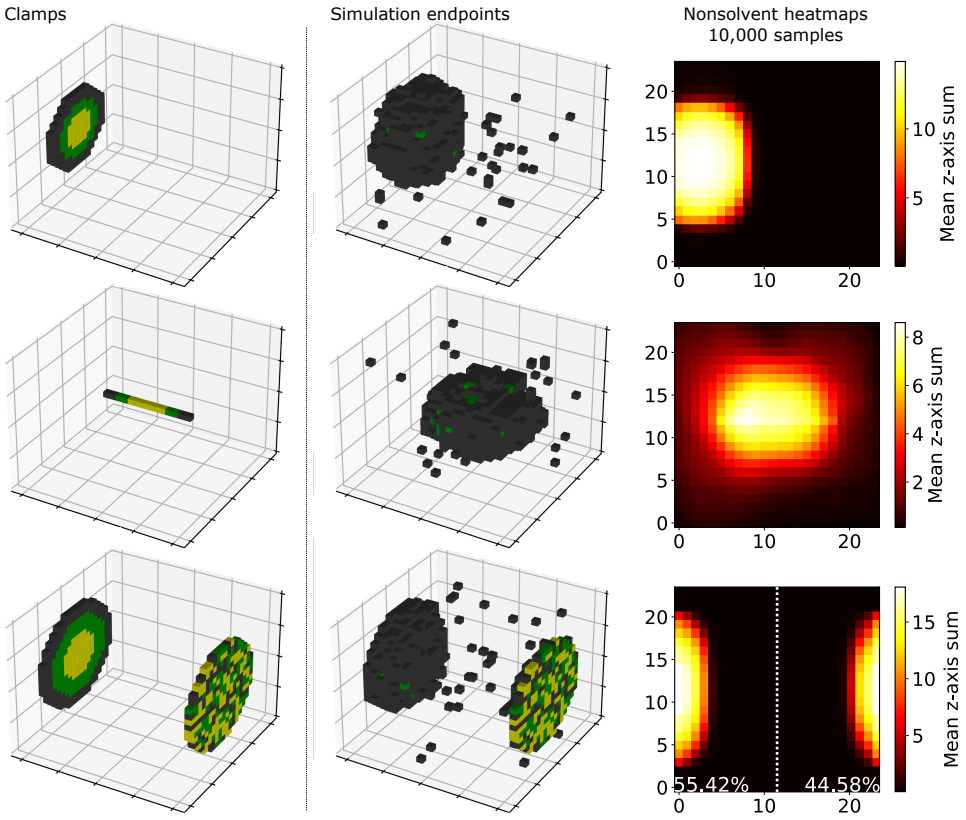

Figure 2 Various examples of inference via clamping, demonstrating generalization beyond the training set. Each row corresponds to a canonical simulation with species counts equal to the ideal avocado from Figure 1. Plots on the left show which positions were clamped, and to which molecules. The middle column shows endpoint snapshots of the simulations. On the right are heatmaps of nonsolvent species, collected over 100 independent MCMC trials with 100 samples from each trial. In the bottom right heatmap, percentages of mass in each half of the heatmap are shown.

in decision making: the spatial distribution of the molecules favors the surface which matches the nested avocado structure. The ability for the ensemble to differentiate between the molecular arrangements on the clamps suggests the capability for differentiation of more complex spatial heterogeneity, which we explore in Section 7.

6 Hopfield droplets: condensation of nonorthogonal, multicomponent droplets conditioned on surface composition

In the cell, stress granules and P-bodies are condensates of differing composition which play distinct roles in mRNA metabolism [20]. However, some molecules, such as proteins like TIA-1, G3BP, and some small RNAs, are found in both condensates, highlighting the role of separation into nonorthogonal mixtures with some distinct and some shared species. The capacity of phase separation to support such mixtures, and the analogy with Hopfield's associative memories, suggest the term "Hopfield droplet" for this phenomenon [45]. Here we explore the Boltzmann liquid learning rule's ability to carve energy minima for condensation of nonorthogonal compositions, in analogy to Hopfield networks that carve energy minima for a given set of nonorthogonal binary vectors.

We trained an example system with 16 species. Two "memories" were chosen, which describe the composition of the condensates to be trained by the learning algorithm. The memories are binary vectors which indicate whether species i is in the composition or not. The memories were generated at random, with the caveat that we tested their performance in an ideal Hopfield network, since we speculate that good performance in a Hebbian-trained Hopfield network is a precondition for good performance in nonorthogonal condensate formation. (Hopfield networks with Hebbian learning, especially with only 16 nodes, are limited in their capacity to dig minima for nonorthogonal vectors; they can store reliable minima for approximately $0.15N$ uniform random memories, where N is the number of nodes [15, 2]. The connection between Hopfield network capacity and the number of nonorthogonal compositions in condensate formation remains to be shown.)

The simulations were done in the grand canonical ensemble. In these simulations, we kept the G_i parameters fixed (G_0 for the solvent was set to 0, and all other molecules had $G_i = 8$, effectively reducing the ambient concentration of the nonsolvent species). While our proof shows that our gradient descent can also optimize the G_i parameters for the target distribution, we found it empirically difficult compared to learning with fixed G_i's. In Appendix A.2, we show a similar result in the canonical ensemble, with different considerations given the ensemble differences.

In the wake phase of training, values of $n_{i,j}$ were calculated for random samples of well-mixed droplets pressed against the wall of the lattice, consisting of uniform composition of species from one memory or the other, as shown in Figure 3a. In the sleep phase, a surface was clamped to one of two memory compositions as shown in Figure 3b. After training, we found that clamping a surface to a memory's composition recruited that composition as a droplet onto the surface, with statistical evidence shown in Figure 4b.

As shown in Figure 4, in analogy to partial recall in Hopfield networks, we showed that a surface with a composition consisting of only a (randomly chosen) subset of a memory can recall the full composition from that memory. In addition, Figure 4 also shows a "polymer" clamped in space, with the composition of a subset of a memory. This polymer clamp also condensed the full memory, showing that the trained parameters are not restricted to forming surface droplets. An extension of the polymer was made with an "inert" molecule, showing that condensation was localized, perhaps analogous to coactivator condensation in super-enhancers for gene regulation [32]. (The "inert" molecule had energies $G_{i,j} = 1$ for all interaction energies and $G_i = 8$.) Partial recall and the alternative clamping style are further examples of a trained model generalizing beyond its training set, as in the various avocado clamps in Figure 2.

Lastly, as shown in Figure 4b, the absence of any clamped voxels yields no condensation. This was trained for implicitly via the training set, since the wall opposite the clamped droplet is trained not to condense any molecules. Thus we show that the trained energies yield condensation only if some voxels (a surface or otherwise) are clamped to provide energy to allow condensation. That is, for grand canonical ensembles of multicomponent liquids, there is an analog to dew point, such that slightly above the dew point, finite-sized condensation occurs as a response to the matching species composition – a form of pattern recognition.

7 Learning structural pattern recognition

In this section we train a set of molecules to recognize structural surface patterns, in contrast to the compositional surface patterns of the Hopfield droplets from Section 6. In addition, this section illustrates the learning of parameters for "hidden" species, whose behavior are not specified by the target distribution.

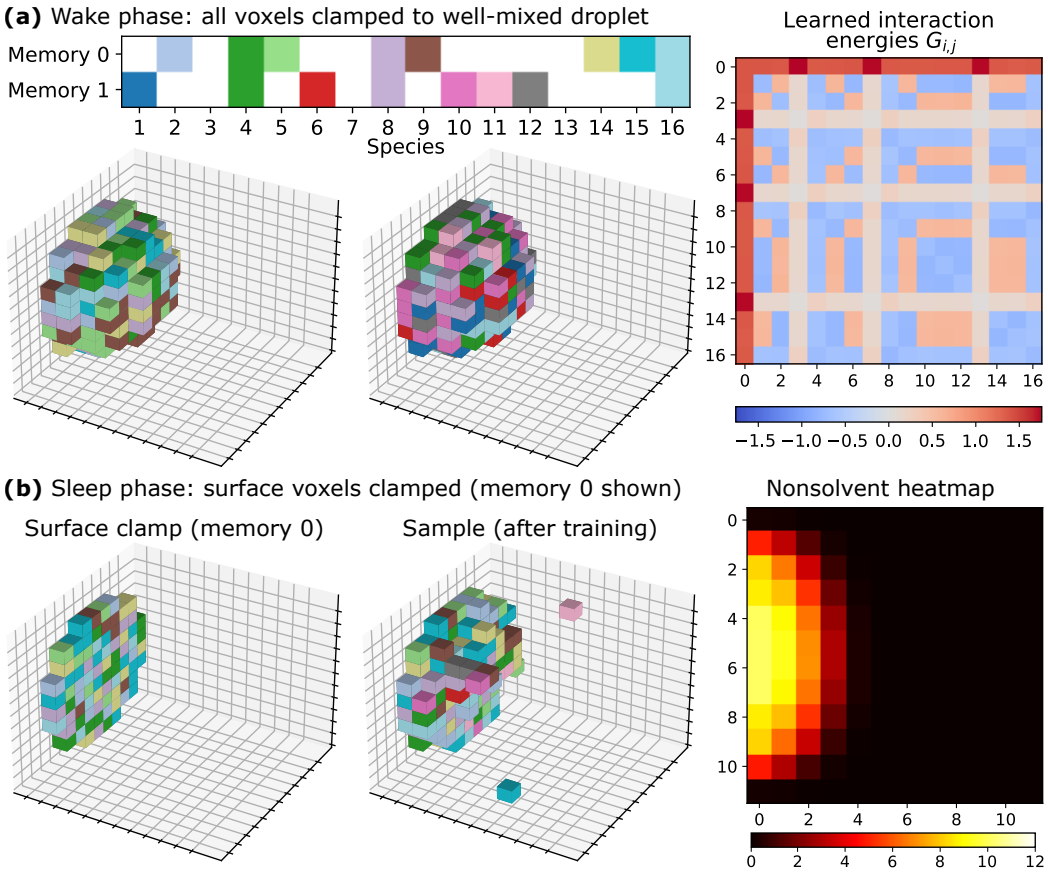

Figure 3 Hopfield droplets: training of droplet condensation of nonorthogonal compositions. Which composition is formed is controlled by the composition of the surface clamp. **(a)** Two "memories", or subsets of the sixteen species, were selected to be learned by the molecular interactions. Some species are unique to a memory, some are shared between both memories, and some are not used by either memory. **(b)** On the left is an example surface clamp used in the sleep phase of training for memory 0. In the middle, a grand canonical simulation endpoint using the trained parameters is shown. On the right is a heatmap, collected from 1,000 MCMC samples, showing the average shape of the droplet.

The task was set up as shown in Figure 5. The simulations were canonical. To ensure that the pattern recognition was not due to surface composition, we chose two surfaces with the same composition but different molecular arrangements. The surfaces consisted of two species in one of two patterns: a "checkers" or "stripes" surface. Recognition was considered successful if one chosen "recognition" species preferred the checkers surface, and another chosen recognition species preferred the stripes surface. We did not expect this task to be achievable with only four species (and solvent), so we included four extra "hidden" species which would form an interface between the surfaces and the recognition species.

We found that training was difficult, but was aided by annealing. We suspect this is because the training algorithm produced stronger weights than needed for other tasks, suggesting that this system is more like crystalline self-assembly than liquid phase.

After training, we found that the recognition was achieved, and the hidden species aided the task in an interpretable way: two species are dedicated to detecting the checkers pattern, by having favorable interactions between each other, the surface, and the checkers recognition

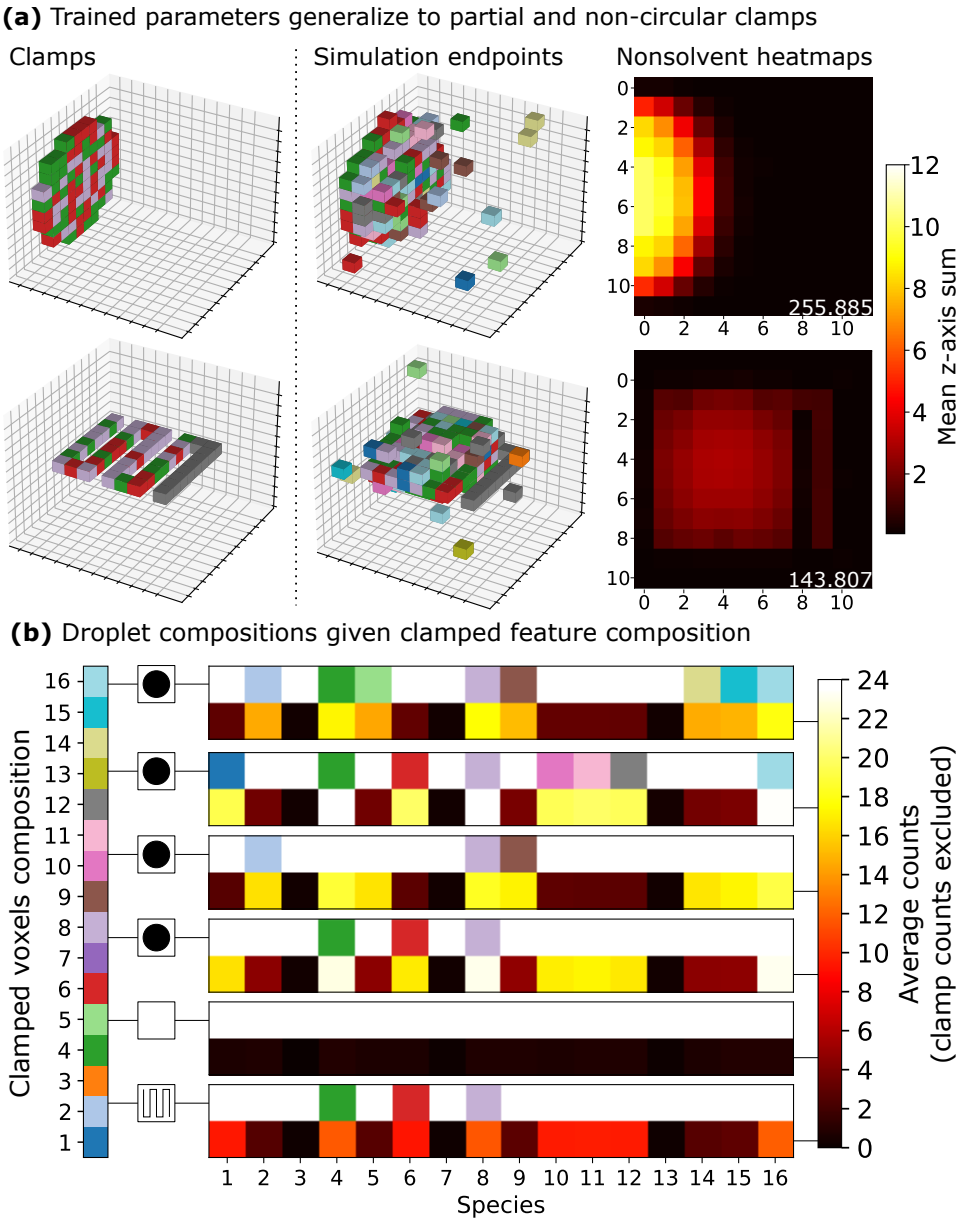

Figure 4 The trained Hopfield droplet system generalizes beyond the training set. **(a)** Plots in the middle are simulation endpoints given the clamped voxels shown on the left. An extra region of the polymer clamp consisted of an extra species (gray) which was "inert". The values in the lower-right of each nonsolvent heatmap are the sums of the heatmap, indicating that the polymer clamp condensed a smaller condensate on average. **(b)** Clamped voxel composition (top row) condensed a droplet with average counts shown in the bottom row, collected from 1,000 MCMC samples. Average counts shown exclude the clamped voxel counts. The squares to the left of each plot show the surface type: a circular surface, no clamp (which did not condense any droplet), and the polymer clamp.

species. The two other hidden species recognized the stripes pattern, by having strong self-affinity (in contrast to strong inter-affinity in the checkers case). We propose that without this extra layer in between the clamped surface and recognition species, this task could not be achieved, suggesting the need for hidden species. This example illustrates the ability of

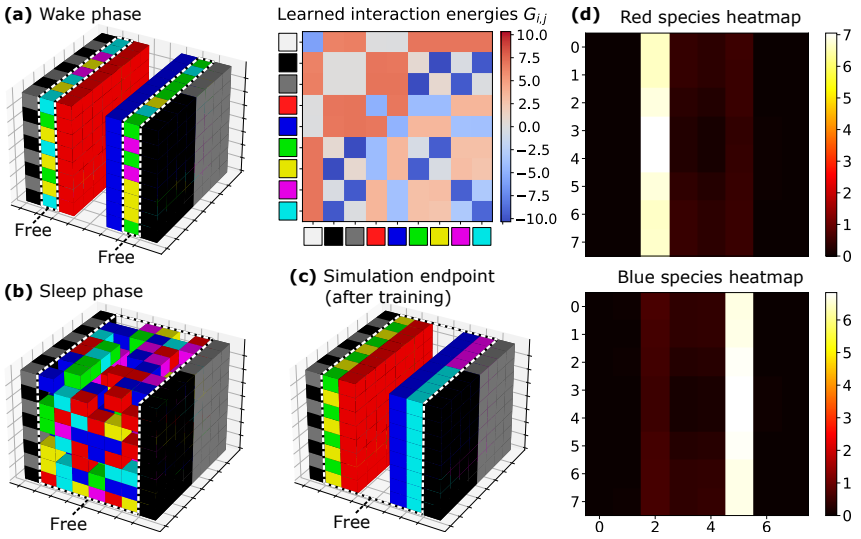

Figure 5 Training recognition of structural surface patterns in a canonical ensemble. Two surface patterns ("checkers" and "stripes" patterns) were provided and the system was trained to localize one "recognition" molecule type (red) to the checkers surface, and the other (blue) to the stripes surface. **(a)** The wake phase consisted of clamping the voxels not labeled free in the plot. (Recall that we allow long-range swaps, so the molecules can move freely between the free regions.) **(b)** In the sleep phase, only the surfaces were clamped. **(c)** A configuration sampled from the end of a (canonical) simulation shows that the species clamped during wake phase (gray, black, red, blue, and solvent) but free in this simulation match the target distribution. **(d)** Heatmaps of 10,000 MCMC samples (each from independent trials) indicating that the red and blue recognition molecules tended to attach to their respective surfaces on average.

the learning rule to train parameters for species whose behavior is unspecified by the training distribution, as well as the ability for our model to recognize structural patterns in contrast to compositional ones.

8 Learning and inference of molecular counts

To illustrate the generality of our learning rule, we show an alternative to clamping voxels in the lattice. The example we used to illustrate this was the observable of counts of each species in the lattice. To explore the space of count vectors, we needed to simulate the grand canonical ensemble, because in the canonical ensemble counts of species are fixed. In this context, inference happens of the form: what is the distribution of counts of molecules a, b, c conditioned on molecule counts x, y, z being fixed? In our MCMC equilibrium sampling, we used a hybrid method in which a proposal was either the swapping of a molecule type in a single position with another molecule type (grand canonical), or swapping of two molecules' positions within the lattice (canonical). Given a set of "clamped" species, we rejected Metropolis-Hastings proposals which attempted to change counts of those species. The hybrid method was required to maintain state-space reachability since clamped species in the lattice would never change their position without position-swap moves. The clamping of a subset of species' counts in the lattice is best likened to semipermeable membranes in biology (sketched in Figure 6a), which regulate the transport of various substances across compartments via permeability criteria like size or charge. Permeability can be changed based on environmental factors, like ion channels changing their permeability based on voltage changes, ligands, or mechanical forces.

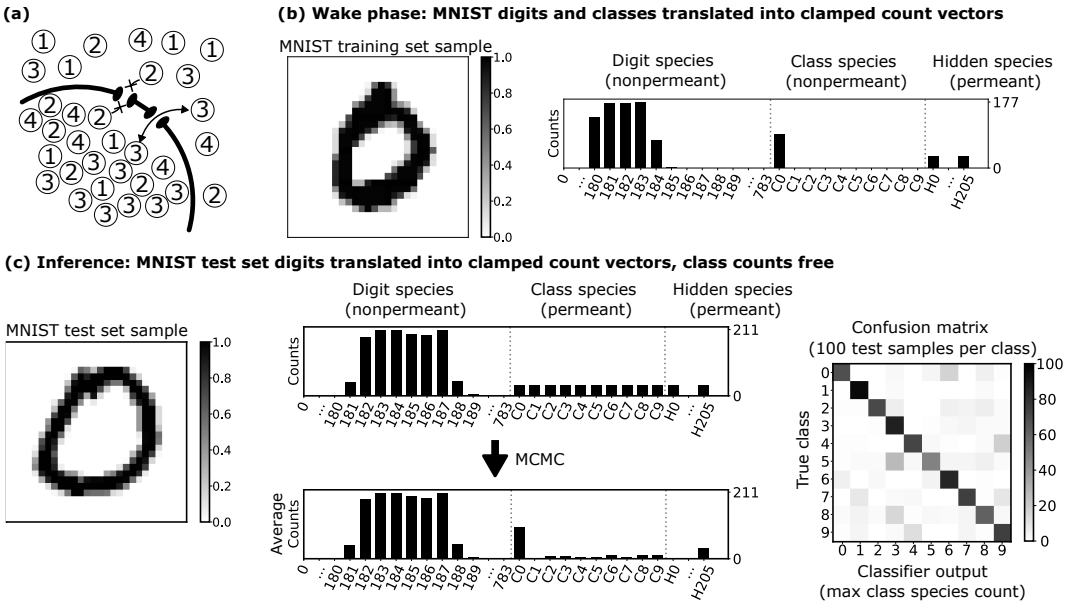

Figure 6 Clamping molecular counts in a "semipermeable membrane"-like grand canonical simulation yields energies which classify clamped count vectors derived from MNIST digits into recruitment of appropriate class species. **(a)** A cartoon representation of a semipermeable membrane, through which molecule type 3 is permeant and molecule type 2 is nonpermeant. **(b)** MNIST digits from the training set (28×28 values in $[0, 1]$ with associated class label) were mapped to species counts in the lattice. The initial counts for the wake phase are shown to the right, along with which molecules were allowed to perform grand canonical swaps in the simulation. **(c)** An example test after training. Clamped to the digit counts given the test digit on the left, MCMC was used to calculate the average equilibrium counts shown at the bottom. On the right, running 100 tests per class and using the maximum of the average class counts as output, each row indicates what the true class was and how many of the 100 tests output each label.

We mapped digits from the MNIST database of handwritten digits into vectors of counts of species, with the aim of training the interactions between the species to recognize which digit was presented. In total, there were 1,000 species. There were 784 digit species, since MNIST digits are provided as 28×28 arrays of values 0 to 1. There were 10 class species, one for each digit class zero through nine. To illustrate the use of hidden species, we added 206 hidden species, which were unclamped during all training and inference.

During the wake phase of training, a random MNIST sample was drawn and the digit and class species were initialized and then clamped. The digit species were each initialized with counts proportional to the value of their pixel in the sampled MNIST digit image, with a total volume fraction of approximately 78.4%. The true (correct) class count was initialized to 1% of the volume, and all incorrect class counts were initialized to 0. The hidden species were initialized uniformly in the remaining 20.6%. The lattice was then shuffled (to a uniform random permutation), and then MCMC sampling calculated $\langle n_{i,j} \rangle$ with the digit and class species clamped. During the sleep phase, a configuration was sampled uniformly at random (without respect to the counts used in training), and MCMC sampling calculated $\langle n_{i,j} \rangle$ with no species clamped. The learned parameters are not shown due to space constraints. The $G_{i,j}$ parameters ranged approximately from -3.5 to 2, while the G_i parameters ranged approximately from -1.2 to 1.3.

During inference, class counts were initialized uniformly, digit counts initialized in the same way as wake training (except using digits sampled from the MNIST test set), and hidden counts initialized uniformly. Then, MCMC simulations were run with only the digit

species clamped. Figures 6b and 6c show example initial counts, and an example of average counts at equilibrium approximated via MCMC. In this simulation, we did not include an explicit solvent species. Inference occurred as the lattice swapped out incorrect class species and recruited correct class species. After training, using the maximum average class count as the output, the set of learned energies enabled 75.4% classification accuracy as shown by the confusion matrix in Figure 6c. (We do not expect we have reached the limit of the accuracy of this method, i.e., longer training or other optimizations such as improved learning algorithms [37] may improve the result.) Thus, we have shown an ability for inference to occur in the compositional rearrangement of a collection of molecules conditioned on certain counts in the collection being fixed.

9 Discussion

Our work illustrates how principles of neural computation, and specifically the Boltzmann machine architecture, shed new light on the capabilities of multicomponent liquids. The Boltzmann liquid model, which augments a standard lattice model with notions of clamping as well as a wake/sleep Hebbian/anti-Hebbian learning rule, provides an interpretation of molecular behavior as representing a high-dimensional probability distribution that can be queried by clamping and tuned by learning. We illustrated this with examples of structures (an avocado), associative memory (Hopfield droplets), geometric pattern recognition (checkers and stripes), and informational pattern recognition (MNIST).

From a machine learning perspective, it is not yet clear how powerful the Boltzmann liquid model is. Unlike classical Boltzmann machines, we do not (yet) have a universal approximation theorem that says with enough hidden units, any probability distribution can be approximated arbitrarily well. Any such theorem would have to account for the fact that the energy model does not distinguish certain symmetries, for example. Understanding contexts (e.g. canonical vs grand-canonical, representations for encoding information) in which Boltzmann liquids perform well is an important question for future work.

For example, there may be more natural ways for Boltzmann liquids to solve the MNIST digit recognition problem. We conjecture that it (and similar inferential problems) can be formulated as a selective surface condensation problem, as in our avocado (Section 5) and Hopfield memory (Section 6) examples. In this formulation, successful inference would be free-floating digit species proportional to an MNIST digit zero condensing onto a surface consisting of "class zero", and not onto other class' surfaces. A more challenging alternative would be to have just two "input" species, black and white, and ask the liquid to distinguish the geometric pattern (the digit image) in a surface clamp, as in our checkers-and-stripes (Section 7) example. The larger question is "what kinds of inference and conditional behaviors are natural for multicomponent liquids?"

Some of the behaviors we demonstrated with Boltzmann liquids have previously been shown in other models for multicomponent liquids. Mao et al [24, 23] used continuous Flory-Huggins and Cahn-Hilliard models to design interaction matrices that give rise to core-shell structures (similar to our avocado) and other geometries; our work provides a distinct learning approach toward the same end. Teixeira et al [45] explore associative recall similar to our Hopfield droplets; critically, their model generalizes Flory-Huggins energies to include a cubic self-repulsion term that they argue is necessary for stabilizing multiple memories. That our discrete model demonstrates multiple memories and associative recall without the cubic terms suggests that our results reflect finite-size effects and may not scale to large droplets where the continuous models are more appropriate. We are not aware of prior work that demonstrates geometrical pattern recognition or informational pattern recognition similar to our checkers-and-stripes and MNIST examples.

Considering our results in the context of biological cells, we imagine that parameters $G_{i,j}$ and G_i provide a continuous space for cells to tune condensate formation and possibly information processing. Intermolecular interactions ($G_{i,j}$) are modulated at multiple time-scales, during and across the cell-cycle through post-translational modifications [31, 42] and over evolutionary time-scales through changing sequence features, and thus their interactions. Molecular counts, and thus G_i, can also be tuned through regulation of gene expression. For example, RNA lengths and sequence determine their electrostatic properties, suggesting a role for non-coding RNAs as tunable knobs for condensate formation in the cell. Beyond isolated droplets, condensation as inference triggered from surface-based clamps (Figure 2) provide a mechanism for spatially regulating condensate assembly and function. Multiple cellular surfaces may exploit similar mechanisms, by assembling gene regulatory condensates only at specific genomic loci [41, 25] and localizing condensates to membranes [42].

Phase-separated biomolecular condensates often involve more complex molecules for which the idealized isotropic cubes of the Boltzmann liquid model are a poor match, which raises the question of how the model can be generalized. Accommodating anisotropic molecules, molecular shapes that span multiple lattice sites, orientation and conformation changes, and local non-nearest-neighbor interactions should be possible with almost trivial changes to the energy model, the move set, and the wake/sleep learning algorithm – but at the cost, most likely, of increased MCMC equilibration time. As this generalization would encompass important features of other models of molecular self-organization, including sticker-and-spacer polymer phase separation [6], DNA tile self-assembly [10, 11], and lattice protein folding [34], we expect that considerably increased capability for representing complex probability distributions would result, with concomitant enhanced potential for sophisticated structural and informational behaviors.

References

1 David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive Science*, 9(1):147–169, 1985.

2 Daniel J Amit, Hanoch Gutfreund, and Haim Sompolinsky. Storing infinite numbers of patterns in a spin-glass model of neural networks. *Physical Review Letters*, 55(14):1530, 1985.

3 Silvia Biffi, Roberto Cerbino, Francesca Bomboi, Elvezia Maria Paraboschi, Rosanna Asselta, Francesco Sciortino, and Tommaso Bellini. Phase behavior and critical activated dynamics of limited-valence DNA nanostars. *Proceedings of the National Academy of Sciences*, 110(39):15633–15637, 2013.

4 Robert Brijder. Computing with chemical reaction networks: a tutorial. *Natural Computing*, 18:119–137, 2019.

5 Jehoshua Bruck. On the convergence properties of the Hopfield model. *Proceedings of the IEEE*, 78(10):1579–1585, 1990.

6 Jeong-Mo Choi, Alex S Holehouse, and Rohit V Pappu. Physical principles underlying the complex biology of intracellular phase transitions. *Annual Review of Biophysics*, 49(1):107–133, 2020.

7 Matthew Cook, David Soloveichik, Erik Winfree, and Jehoshua Bruck. Programmability of chemical reaction networks. In *Algorithmic Bioprocesses*, pages 543–584. Springer, 2009.

8 David Doty. Theory of algorithmic self-assembly. *Communications of the ACM*, 55(12):78–88, 2012.

9 John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(7), 2011.

10 Constantine G Evans and Erik Winfree. Physical principles for DNA tile self-assembly. *Chemical Society Reviews*, 46(12):3808–3829, 2017.

11 Constantine Glen Evans, Jackson O'Brien, Erik Winfree, and Arvind Murugan. Pattern recognition in the nucleation kinetics of non-equilibrium self-assembly. *Nature*, 625(7995):500–507, 2024.

12 Marina Feric, Nilesh Vaidya, Tyler S Harmon, Diana M Mitrea, Lian Zhu, Tiffany M Richardson, Richard W Kriwacki, Rohit V Pappu, and Clifford P Brangwynne. Coexisting liquid phases underlie nucleolar subcompartments. *Cell*, 165(7):1686–1697, 2016.

13 Archa H Fox, Shinichi Nakagawa, Tetsuro Hirose, and Charles S Bond. Paraspeckles: where long noncoding RNA meets phase separation. *Trends in Biochemical Sciences*, 43(2):124–135, 2018.

14 Geoffrey E Hinton and Terrence J Sejnowski. Analyzing cooperative computation. In *Proceedings of the Fifth Annual Conference of the Cognitive Science Society*, pages 2554–2558, 1983.

15 John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.

16 John J Hopfield, David I Feinstein, and Richard G Palmer. Unlearning has a stabilizing effect in collective memories. *Nature*, 304(5922):158–159, 1983.

17 William M Jacobs and Daan Frenkel. Predicting phase behavior in multicomponent mixtures. *The Journal of Chemical Physics*, 139(2), 2013.

18 William M Jacobs and Daan Frenkel. Phase transitions in biological systems with many components. *Biophysical Journal*, 112(4):683–691, 2017.

19 Byoung-jin Jeon, Dan T Nguyen, and Omar A Saleh. Sequence-controlled adhesion and microemulsification in a two-phase system of DNA liquid droplets. *The Journal of Physical Chemistry B*, 124(40):8888–8895, 2020.

20 Nancy Kedersha and Paul Anderson. Mammalian stress granules and processing bodies. *Methods in Enzymology*, 431:61–81, 2007.

21 Tsung-Dao Lee and Chen-Ning Yang. Statistical theory of equations of state and phase transitions. II. Lattice gas and Ising model. *Physical Review*, 87(3):410, 1952.

22 Andrew S Lyon, William B Peeples, and Michael K Rosen. A framework for understanding the functions of biomolecular condensates across scales. *Nature Reviews Molecular Cell Biology*, 22(3):215–235, 2021.

23 Sheng Mao, Milena S Chakraverti-Wuerthwein, Hunter Gaudio, and Andrej Košmrlj. Designing the morphology of separated phases in multicomponent liquid mixtures. *Physical Review Letters*, 125(21):218003, 2020.

24 Sheng Mao, Derek Kuldinow, Mikko P Haataja, and Andrej Košmrlj. Phase behavior and morphology of multicomponent liquid mixtures. *Soft Matter*, 15(6):1297–1311, 2019.

25 Jose A Morin, Sina Wittmann, Sandeep Choubey, Adam Klosin, Stefan Golfier, Anthony A Hyman, Frank Jülicher, and Stephan W Grill. Sequence-dependent surface condensation of a pioneer transcription factor on DNA. *Nature Physics*, 18(3):271–276, 2022.

26 Satoshi Murata. *Molecular Robotics: An Introduction*. Springer Nature, 2022.

27 Arvind Murugan, Zorana Zeravcic, Michael P Brenner, and Stanislas Leibler. Multifarious assembly mixtures: Systems allowing retrieval of diverse stored structures. *Proceedings of the National Academy of Sciences*, 112(1):54–59, 2015.

28 Matthew J Patitz. An introduction to tile-based self-assembly and a survey of recent results. *Natural Computing*, 13:195–224, 2014.

29 William Poole, Andrés Ortiz-Munoz, Abhishek Behera, Nick S Jones, Thomas E Ouldridge, Erik Winfree, and Manoj Gopalkrishnan. Chemical Boltzmann machines. In *DNA Computing and Molecular Programming (DNA23)*, pages 210–231. Springer, 2017.

30 William Poole, Thomas Ouldridge, Manoj Gopalkrishnan, and Erik Winfree. Detailed balanced chemical reaction networks as generalized Boltzmann machines. *arXiv preprint arXiv:2205.06313*, 2022.

31 Arpan Kumar Rai, Jia-Xuan Chen, Matthias Selbach, and Lucas Pelkmans. Kinase-controlled phase transition of membraneless organelles in mitosis. *Nature*, 559(7713):211–216, 2018.

32 Benjamin R Sabari, Alessandra Dall'Agnese, Ann Boija, Isaac A Klein, Eliot L Coffey, Krishna Shrinivas, Brian J Abraham, Nancy M Hannett, Alicia V Zamudio, John C Manteiga, et al. Coactivator condensation at super-enhancers links phase separation and gene control. *Science*, 361(6400):eaar3958, 2018.

33 Ruslan Salakhutdinov and Geoffrey Hinton. Deep Boltzmann machines. In *Artificial Intelligence and Statistics*, pages 448–455. PMLR, 2009.

34 Andrej Šali, Eugene Shakhnovich, and Martin Karplus. Kinetics of protein folding: A lattice model study of the requirements for folding to the native state. *Journal of Molecular Biology*, 235(5):1614–1636, 1994.

35 Yusuke Sato, Tetsuro Sakamoto, and Masahiro Takinoue. Sequence-based engineering of dynamic functions of micrometer-sized DNA droplets. *Science Advances*, 6(23):eaba3471, 2020.

36 John E Savage. *Models of Computation: Exploring the Power of Computing*. Addison-Wesley Reading, 1998.

37 Benjamin Scellier and Yoshua Bengio. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in Computational Neuroscience*, 11:24, 2017.

38 Ludovica Serricchio, Dario Bocchi, Claudio Chilin, Raffaele Marino, Matteo Negri, Chiara Cammarota, and Federico Ricci-Tersenghi. Daydreaming Hopfield networks and their surprising effectiveness on correlated data. *arXiv preprint arXiv:2405.08777*, 2024.

39 Krishna Shrinivas and Michael P Brenner. Phase separation in fluids with many interacting components. *Proceedings of the National Academy of Sciences*, 118(45):e2108551118, 2021.

40 Krishna Shrinivas and Michael P Brenner. Multiphase coexistence capacity in complex fluids. *bioRxiv*, pages 2022–10, 2022.

41 Krishna Shrinivas, Benjamin R Sabari, Eliot L Coffey, Isaac A Klein, Ann Boija, Alicia V Zamudio, Jurian Schuijers, Nancy M Hannett, Phillip A Sharp, Richard A Young, et al. Enhancer features that drive formation of transcriptional condensates. *Molecular Cell*, 75(3):549–561, 2019.

42 Wilton T Snead and Amy S Gladfelter. The control centers of biomolecular phase separation: how membrane surfaces, PTMs, and active processes regulate condensation. *Molecular Cell*, 76(2):295–305, 2019.

43 Héctor J Sussmann. Learning algorithms for Boltzmann machines. In *Proceedings of the 27th IEEE Conference on Decision and Control*, pages 786–791. IEEE, 1988.

44 Masahiro Takinoue. DNA droplets for intelligent and dynamical artificial cells: from the viewpoint of computation and non-equilibrium systems. *Interface Focus*, 13(5):20230021, 2023.

45 Rodrigo Braz Teixeira, Giorgio Carugno, Izaak Neri, and Pablo Sartori. Liquid Hopfield model: retrieval and localization in heterogeneous liquid mixtures. *arXiv preprint arXiv:2310.18853*, 2023.

46 Hirotake Udono, Jing Gong, Yusuke Sato, and Masahiro Takinoue. DNA droplets: intelligent, dynamic fluid. *Advanced Biology*, 7(3):2200180, 2023.

47 Laurent Younes. Synchronous Boltzmann machines can be universal approximators. *Applied Mathematics Letters*, 9(3):109–113, 1996.

48 Weishun Zhong, David J Schwab, and Arvind Murugan. Associative pattern recognition through macro-molecular self-assembly. *Journal of Statistical Physics*, 167:806–826, 2017.

A Appendix A

A.1 Methods

Simulations utilized GPU accelerated parallel algorithms for the canonical swap and grand canonical replacement update proposals discussed in Section 2. In each simulation step, GPU threads were assigned lattice positions spaced modulo four and were synchronized each step so as to avoid race conditions involving calculation of the energy in the von Neumann neighborhood.

The training in Sections 5, 7, 8, and A.2 utilized AdaGrad, or the adapative gradient algorithm [9]. AdaGrad adapts the learning rates of all parameters independently by scaling them inversely proportional to the square root of the sum of all their historical squared gradients, thus reducing the learning rate for frequently updated parameters. We found AdaGrad reduced the number of training steps required by allowing larger learning rates.

For all MCMC trials, the first half of the simulation steps were not sampled, typically called a *burn-in* period, followed by taking 100 samples at equally spaced steps. Energies were also sampled and checked to have stabilized, i.e., converged to a minima after a burn-in period. Figure 7 shows an example avocado simulation along with its energy plotted against simulation step.

Lattices were always cubic with the following lengths. Sections 5 and A.2: 24, Section 6: 12, Section 7: 8, Section 8: 32.

Each MCMC trial consisted of the following number of parallel steps. Sections 5 and A.2: $4(24^4)$, Section 6: $40(12^4)$, Section 7: $10(8^4)$, Section 8: $20,000$. Each parallel step proposed $L^3/4^3$ swaps, where L was the lattice length.

The learning rates for Euler integration were as follows. Sections 5 and A.2: $5,000/24^3$, Section 6: $1/12^3$, Section 7: $50/8^3$, Section 8: 0.025.

The total number of Euler integration steps were as follows. Section 5: 80, Section A.2: 500, Section 6: $3,590$, Section 7: $1,741$, Section 8: $2,203$, with each step consisting of 10 MCMC trials, each sampling a digit from a different class of the MNIST training set.

In Section 7, annealing was performed only during the burn-in period with $kT_{\text{high}} = 10$ using $kT(i) = kT_{\text{high}} \cdot \left(\frac{1}{kT_{\text{high}}}\right)^{\frac{i}{\text{burn-in steps}}}$ for step i.

A.2 Hopfield droplets in the canonical ensemble

In this section, we describe a framework for studying associative recall and other Hopfield network phenomena in a canonical ensemble of surface-conditioned droplet formation.

Two key points determine our choice of counts for each species in the canonical ensemble. First, for nonorthogonal memories, it is important to show that compositions can form even in the presence of all species, since some species in the composition will have favorable energies with species outside the composition. Secondly, due to the overlap in compositions, we expected the species outside the memory to have enough affinity to attach to the surface droplet, thus forming one large droplet of both compositions. To address these issues, we included compositions called anti-memories, which are the complements of the desired memories. Thus the targeted separation is into memory 0 and anti-memory 0, or memory 1 and anti-memory 1.

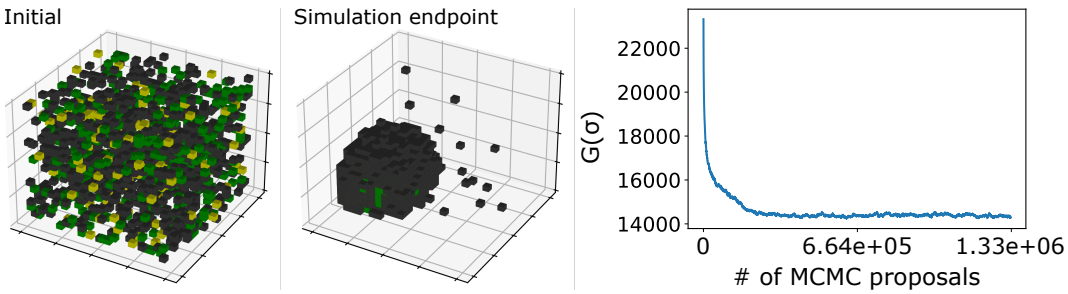

Figure 7 A canonical simulation of the avocado system described in Section 5. On the left, the initial state, which is a random permutation of the ideal avocado. In the middle, the endpoint of the simulation is shown. On the right, a plot of the energy of the lattice configuration during simulation.

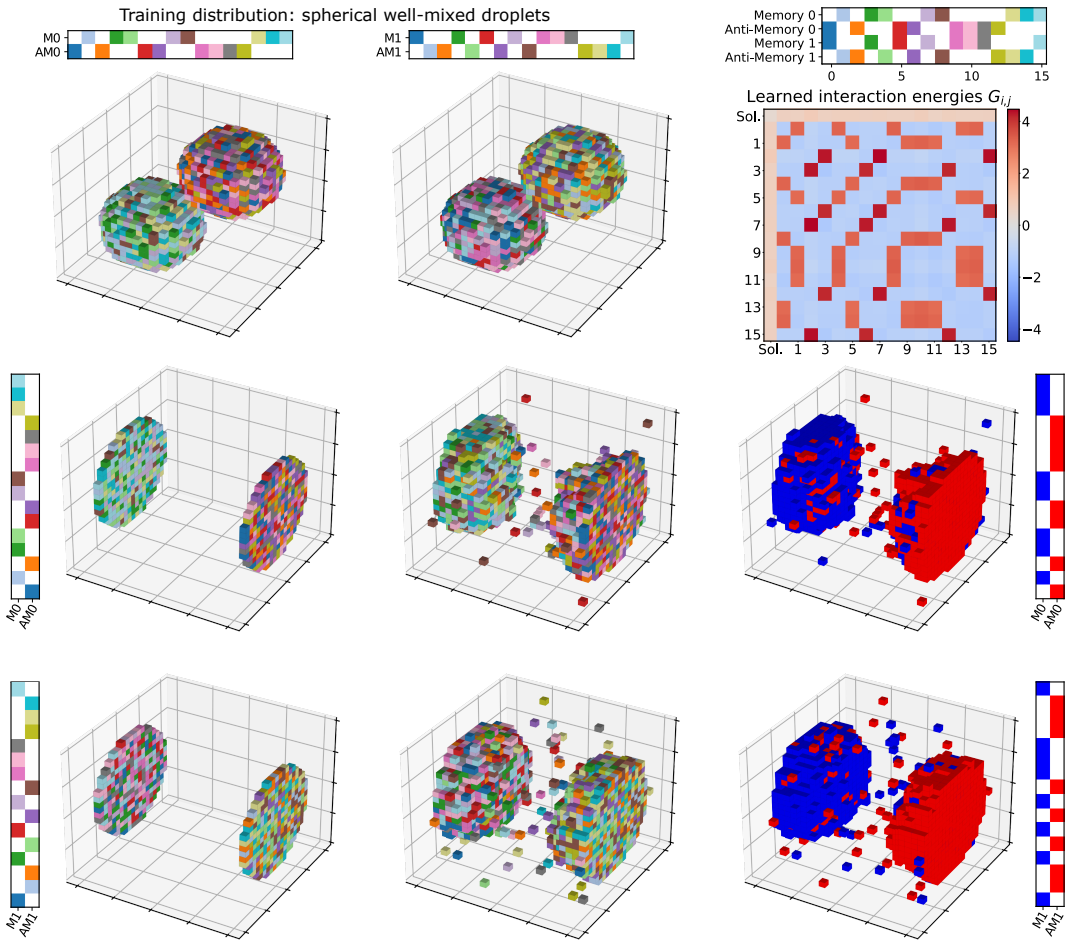

Figure 8 Learning of multicomponent, nonorthogonal condensation in a canonical ensemble. The rectangles with colored squares indicate which species are in which memory and anti-memory. The top shows two samples from the training distribution. In the middle and bottom row, the left plots show the surface clamps. In the middle, simulation endpoints are shown. The plots on the right are filtered such that the desired memory species are colored red, and the desired anti-memory species are colored blue. Statistical evidence from MCMC sampling is shown in Figures 10.

As shown in Figure 8, training on droplets consisting of ideal spheres of a set of memory/anti-memory pairs, surrounded by solvent, yields interaction energies where one memory/anti-memory pair can be selectively recruited based on surface clamping. The training procedure reinforces strengths between species which are shared in any of the four compositions, while weakening strengths if species are too correlated in an unclamped simulation. After training, clamping one surface to molecules in a memory's composition and the other surface to molecules in the corresponding anti-memory's composition, the equilibrium distribution (as sampled by MCMC) is conditioned by the clamps to consist of the memory composition and the anti-memory composition cleanly separated, as shown in the heatmaps in Figure 10.

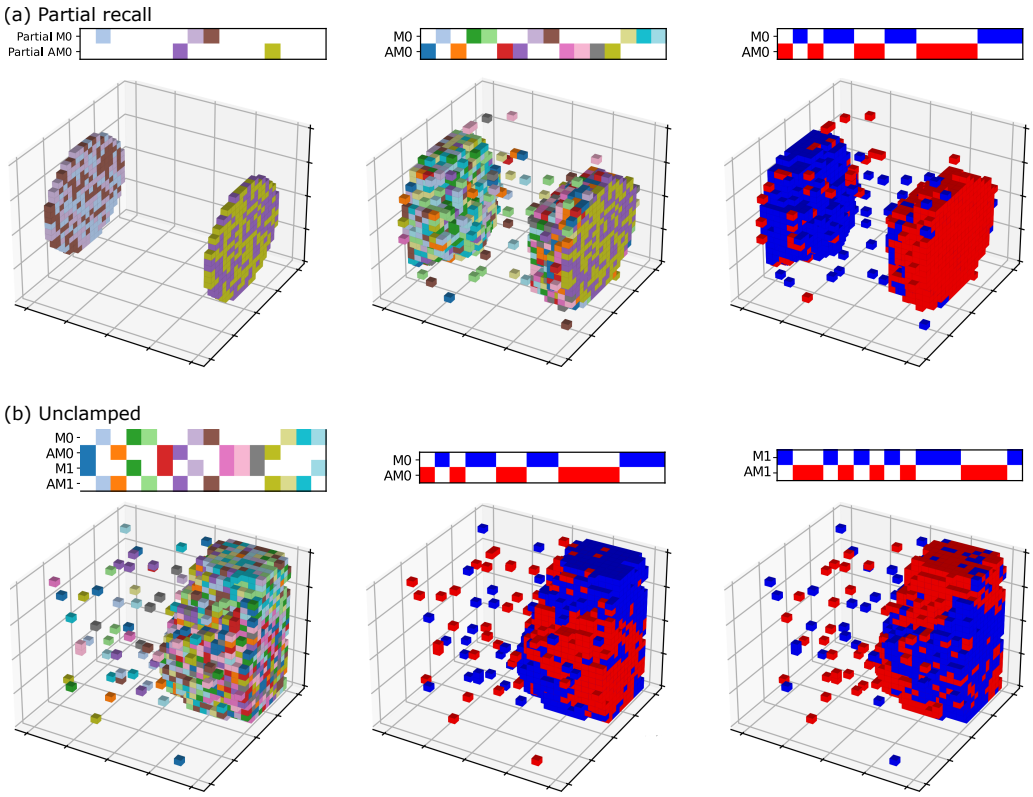

Figure 9 Examples of partial recall, and the inability to separate in the absence of a clamped surface. **(a)** The left shows surface clamps, consisting of the partial memory compositions shown on above the plot. The middle plot shows a simulation endpoint snapshot. In the right plot, blue and red indicate membership in Memory 0 and Anti-Memory 0, respectively. **(b)** A simulation endpoint snapshot without surface clamps. The middle and right plots show a coloring according to memory membership.

In Figure 9a, we show that clamping the surfaces to only a subset of a memory and anti-memory can still be sufficient to condition the probability distribution towards clean separation of the two compositions, analogous to partial recall in Hopfield networks. (Figure 10 provides statistical evidence from MCMC sampling.) Partial clamping suggests that even if nonorthogonal condensates consist of many components, surfaces (or other forms of clamping) consisting of few species can still drive selective separation.

In the absence of clamps, the learned energies do not yield cleanly separated droplets of any composition, and instead form a single condensate as shown in Figure 9b. The inability for the learning rule to find a solution suggests that additional measures may be required to ensure clean droplet separation, for example via the use of surfactants to create membranes, as is common in droplets in biology and chemistry. While one option would be to introduce surfactants in clamped positions surrounding the droplet in the training distribution, we speculate that the inclusion of unclamped "hidden" molecules during training could allow the learning rule to uncover surfactants as a solution to the separation problem without being designed to learn surfactants explicitly.

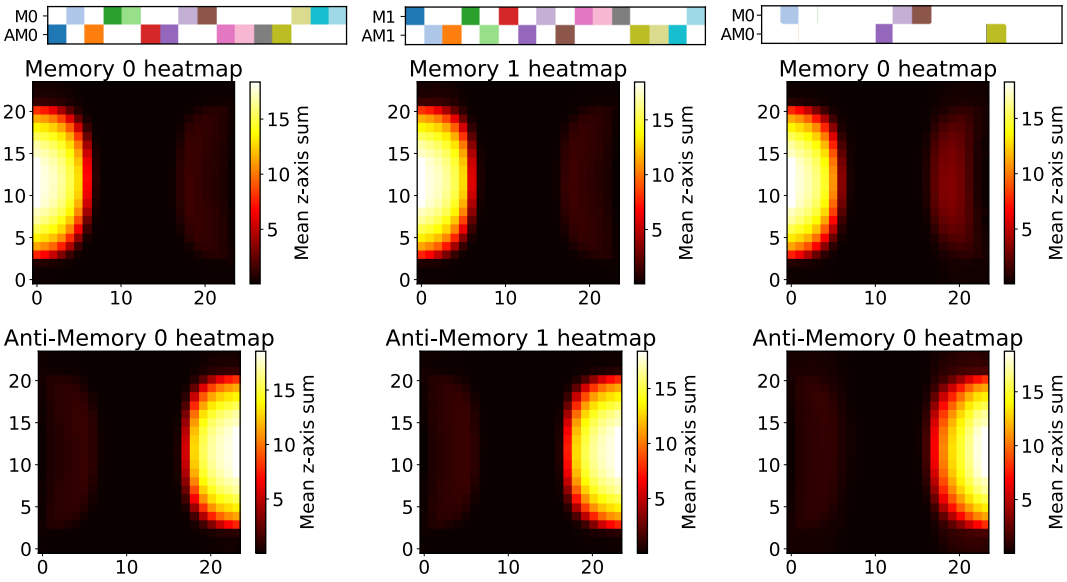

Figure 10 Heatmaps of Figure 8 and Figure 9a. Averages were collected over 100 independent MCMC trials collecting 100 samples each. The heatmaps sum only the presence of species in the memory listed in their title, e.g., the top left heatmap is the average sum through the z-axis of only species from Memory 0. The vector of species listed at the top shows the surface clamps used in the simulation.

A.3 Proof of learning rule

In this derivation, we let $kT = 1$. The derivation below holds when replacing G_{ij} with G_i, suggesting the same weight updates for per-molecule energies G_i in the grand canonical ensemble. By the definition of the Kullback-Leibler divergence, and the fact that Q_a (the provided target distribution) is independent of the model parameters G_{ij}, we have:

$$\frac{\partial D_{\mathrm{KL}}(Q_a \,||\, P_a)}{\partial G_{ij}} = \frac{\partial}{\partial G_{ij}} \sum_{\alpha} Q_a(\alpha) \ln \frac{Q_a(\alpha)}{P_a(\alpha)} \tag{4}$$

$$= -\sum_{\alpha} Q_a(\alpha) \frac{\partial}{\partial G_{ij}} \ln P_a(\alpha). \tag{5}$$

Recall that $P_a(\alpha) = \sum_{\sigma \in M_a(\alpha)} P(\sigma)$, and $P(\sigma) = \frac{e^{-G(\sigma)}}{Z}$, then we have:

$$\frac{\partial}{\partial G_{ij}} \ln P_a(\alpha) = \frac{1}{P_a(\alpha)} \frac{\partial P_a(\alpha)}{\partial G_{ij}} \tag{6}$$

$$= \frac{1}{P_a(\alpha)} \sum_{\sigma \in M_a(\alpha)} \frac{\partial P(\sigma)}{\partial G_{ij}} \tag{7}$$

$$= \frac{1}{P_a(\alpha)} \sum_{\sigma \in M_a(\alpha)} \frac{\partial}{\partial G_{ij}} \frac{e^{-G(\sigma)}}{Z} \tag{8}$$

$$= \frac{1}{P_a(\alpha)} \sum_{\sigma \in M_a(\alpha)} \frac{Z \frac{\partial e^{-G(\sigma)}}{\partial G_{ij}} - e^{-G(\sigma)} \frac{\partial Z}{\partial G_{ij}}}{Z^2}. \tag{9}$$

Handling the partial derivatives in the numerator independently yields:

$$\frac{\partial e^{-G(\sigma)}}{\partial G_{ij}} = \frac{\partial e^{-G(\sigma)}}{\partial(-G(\sigma))}\frac{\partial(-G(\sigma))}{\partial G_{ij}} \tag{10}$$

$$= e^{-G(\sigma)}\frac{\partial(-G(\sigma))}{\partial G_{ij}} \tag{11}$$

$$= -e^{-G(\sigma)} n_{ij}^{\sigma}. \tag{12}$$

$$\frac{\partial Z}{\partial G_{ij}} = \sum_{\sigma}\frac{\partial e^{-G(\sigma)}}{\partial G_{ij}} \tag{13}$$

$$= -\sum_{\sigma} e^{-G(\sigma)} n_{ij}^{\sigma}. \tag{14}$$

$$\frac{\partial}{\partial G_{ij}}\ln P_a(\alpha) = \frac{1}{P_a(\alpha)}\sum_{\sigma\in M_a(\alpha)}\frac{Z\frac{\partial e^{-G(\sigma)}}{\partial G_{ij}} - e^{-G(\sigma)}\frac{\partial Z}{\partial G_{ij}}}{Z^2} \tag{15}$$

$$= \frac{1}{P_a(\alpha)}\sum_{\sigma\in M_a(\alpha)}\frac{-Ze^{-G(\sigma)}n_{ij}^{\sigma} + e^{-G(\sigma)}\sum_{\sigma'}e^{-G(\sigma')}n_{ij}^{\sigma'}}{Z^2} \tag{16}$$

$$= -\sum_{\sigma\in M_a(\alpha)}\frac{1}{P_a(\alpha)}\frac{e^{-G(\sigma)}n_{ij}^{\sigma}}{Z} + \sum_{\sigma\in M_a(\alpha)}\frac{1}{P_a(\alpha)}\frac{e^{-G(\sigma)}\langle n_{ij}\rangle_P}{Z} \tag{17}$$

$$= -\sum_{\sigma\in M_a(\alpha)}\frac{P(\sigma)}{P_a(\alpha)}n_{ij}^{\sigma} + \sum_{\sigma\in M_a(\alpha)}\frac{P(\sigma)}{P_a(\alpha)}\langle n_{ij}\rangle_P \tag{18}$$

$$= -\sum_{\sigma\in M_a(\alpha)}\frac{P(\sigma)}{P_a(\alpha)}n_{ij}^{\sigma} + \langle n_{ij}\rangle_P. \tag{19}$$

Recalling that $Q(\sigma) = \sum_{\alpha} Q_a(\alpha)P_a(\sigma)$ and $P_a(\sigma) = P(\sigma \mid \alpha) = P(\sigma)/P_a(\alpha)$,

$$\frac{\partial D_{\mathrm{KL}}(Q_a \,||\, P_a)}{\partial G_{ij}} = -\sum_{\alpha} Q_a(\alpha)\frac{\partial}{\partial G_{ij}}\ln P_a(\alpha) \tag{20}$$

$$= -\sum_{\alpha} Q_a(\alpha)\left(-\sum_{\sigma\in M_a(\alpha)}\frac{P(\sigma)}{P_a(\alpha)}n_{ij}^{\sigma} + \langle n_{ij}\rangle_P\right) \tag{21}$$

$$= \sum_{\alpha}\sum_{\sigma\in M_a(\alpha)}\frac{P(\sigma)}{P_a(\alpha)}Q_a(\alpha)n_{ij}^{\sigma} - \left(\sum_{\alpha}Q_a(\alpha)\right)\langle n_{ij}\rangle_P \tag{22}$$

$$= \langle n_{ij}\rangle_Q - \langle n_{ij}\rangle_P. \tag{23}$$