Brain computation as fast spiking neural Monte Carlo inference in probabilistic programs

George Matheos[a,b,*], **Andrew D Bolton**[b,c,*], **McCoy Becker**[b], **Cameron Freer**[b], **and Vikash Mansinghka**[b]

[a]UC Berkeley; [b]MIT Probabilistic Computing Project; [c]Harvard; [*]Equal Contribution

This manuscript was compiled on November 30, 2022

How can slow, spiking neurons implement the fast probabilistic inferences needed to explain perception and cognition? Biological neurons are millions of times slower than electronic computers, yet they appear to robustly approximate probabilistic inferences in complex probabilistic programs with many latent variables in real-time. Here we show how biologically realistic, massively parallel assemblies of spiking neurons can perform real-time probabilistic inference. Our approach, based on novel weighted Monte Carlo spiking codes that leverage spike rates as well as coarse spike timing, requires exponentially fewer neurons than standard probabilistic population codes. It also scales to real-time inference via massively parallel hybrids of model-based Monte Carlo and data-driven neural networks, and works for high-dimensional probabilistic programs that previous spiking neural inference architectures do not handle. We illustrate generality by providing neurally mappable implementations of resource-rational variants of Bayesian cognitive models for primate mental physical simulation, human learning of numerical concepts, and 3D prey tracking by larval zebrafish. We also confirm predictions of the spiking neural Monte Carlo theory using empirical data drawn from the hodology, functional neuroanatomy, synaptic physiology, and extracellular spike and field electrophysiology of multiple brain regions and model organisms.

Brain computation | probabilistic programming | Monte Carlo | deep learning | spiking neural networks | probabilistic inference | visual perception | mental simulation | concept learning

This paper addresses two questions: In theory, how can slow, spiking neurons possibly implement the fast approximate probabilistic inferences needed for perception and cognition? And can such a theory predict empirical data from studies of hodology, functional neuroanatomy, synaptic physiology, and extracellular spike and field electrophysiology? This paper introduces new, massively parallel architectures for spiking neural Monte Carlo inference in probabilistic programs with many latent variables, overcoming scaling limitations of previous work on spiking neural inference. It also confirms multiple predictions about fundamental biophysical mechanisms, micro-scale circuits, meso-scale networks, and macro-scale architectures and dynamics using empirical data from multiple brain regions and model organisms.

The idea that everyday perception and cognition relies on probabilistic inference in rich, flexible generative models can be traced back at least as far as Helmholtz (1) and Laplace (2). Probabilistic inference in structured probabilistic models played a central role in multiple generations of artificial intelligence systems (3–5) and computational models of cognition (6). Probabilistic inference, and especially sampling-based, Monte Carlo approximate inference approaches, are also central to reverse-engineering approaches in computational cognitive science, especially the traditions of "resource-rational" analysis of cognitive inference processes (7), and in the "Bayesian brain" (8) or "sampling hypothesis" frameworks (9, 10). Unfortunately, despite the conceptual appeal of this perspective, it has proved difficult to bridge the gap between computational theories of inference and neural representation (11, 12).

Probabilistic programming (13–17) provides a computational formalism for generalizing and scaling implementations of inference in generative models. Probabilistic programs with many latent variables are increasingly central to state-of-the-art architectures for real-time 3D computer vision (17, 18) and theory of mind via inverse planning (19, 20) and also to computational cognitive science (21). Probabilistic programs offer new possibilities for solving problems central to embodied intelligence by integrating data-driven and model-driven modes of inference (22, 23), and support state-of-the-art hybrids of sequential Monte Carlo (16, 24) with variational inference (25, 26). Probabilistic programs can even encode risk-sensitive action selection and decision-theoretic planning (13, 27, 28). Unfortunately, thus far, there have been no spiking neural architectures that can scale to perform real-time, high-quality approximate probabilistic inference in probabilistic programs with many latent variables. There is thus a fundamental gap between computational models of intelligence and biologically realistic models of brain computation.

Significance Statement

Cognitive science, neuroscience, and artificial intelligence have not yielded an integrative theory of how probabilistic inference is implemented in the mind and brain, leaving fundamental gaps between phenomenological, causal, and computational accounts of intelligence. Spiking neural Monte Carlo narrows these gaps, offering a theory for reverse-engineering brain computation that is more computationally general, cognitively realistic, and biologically grounded than artificial neural networks on their own. It gives a unifying explanation of micro-scale, meso-scale, and macro-scale features of neural connectivity, coding, and dynamics. It shows how to automatically construct implementations of a broad class of probabilistic programs that encode Bayesian models, and test their predictions against both behavioral and neural data. Finally, it exposes massive micro-scale, meso-scale, and macro-scale parallelism inherent in probabilistic programming, yielding a new brain-like scaling route for engineering intelligent machines.

G.M. and A.B. and V.M. performed research; C.F. and M.B. assisted with research; V.M. designed and oversaw research; and V.M., A.B., and G.M. wrote the paper.

Please declare any competing interests here.

[1,2]George Matheos and Andrew Bolton contributed equally to this work.

[1, 2, 5]To whom correspondence should be addressed. E-mails: georgematheos@berkeley.edu, andrewdbolton@fas.harvard.edu, and vkm@mit.edu

Consider that inference in probabilistic programs is ordinarily implemented using electronic computers that perform hundreds of millions of instructions per second. These computers are in turn implemented via logic gates that transition billions of times per second. In contrast, biological neurons spike millions of times slower, yet many perceptual inferences require just hundreds of milliseconds, and many cognitive inferences require just seconds. This in turn means that the brain must somehow approximate probabilistic inference using massively parallel circuits that integrate new data without the long sequential chains of operations that can be used in software. Influential feedforward models of visual object recognition have just 5 layers (29).

Artificial neural network (ANN) models can be trained to provide low-latency approximate probabilistic inferences. They have been used to build neurally mappable models of primate vision (30–32) and mental simulation (33), as well as larger-scale models (34) simulation. However, fundamental limitations of ANN models, both as AI technology and as models of visual perception, are also increasingly widely recognized (35–38)

Even proponents of ANN models see fundamental open problems, such as how to account for the role of top-down connections in visual cortex, and therefore the computational interactions between bottom-up, data-driven processing and top-down, model-based feedback (39) This limitation appears related to failures of ANN models in practice. Consider that even state-of-the-art extensions to CNNs and RNNs for visual perception, trained on massive datasets via algorithms that lack biologically plausible implementations, exhibit striking failures that biological vision systems do not (37). These include adversarial examples (40), and also other failures of common sense, such as falsely positing everyday 3D objects floating inexplicably in unoccupied space (18), and failing to miss visually salient objects such as pedestrians, trucks, and emergency vehicles (41, 42) Also, standard ANN models do not reflect many fundamental characteristics of biological neural networks, ranging from the laminar structure of cortex to combinations of dense and sparse coding to widespread oscillations and synchrony.

Inspired in part by these challenges, there is a rich literature on spiking neural architectures and other massively parallel circuit formalisms for probabilistic inference. Prominent examples include probabilistic population codes (43, 44); spiking neural Gibbs samplers suitable for inference in discrete Bayesian network models with sufficiently sparse connectivity (45, 46); spiking neural Bayesian filters that extend probabilistic population codes for real-time tracking (47, 48). However, these previous proposals for spiking probabilistic inference cannot implement state-of-the-art schemes for real-time sequential Monte Carlo inference in complex probabilistic programs. There have also been proposals for stochastic digital circuits for massively parallel, low precision, real-time Monte Carlo (49–51) inference in probabilistic graphical models and non-parametric Bayesian models with tens of thousands of variables; and other, more specialized neural inference schemes. But these stochastic digital circuits do not explain how to perform robust, real-time probabilistic inference using components that are as slow as biological neurons.

This paper introduces spiking neural Monte Carlo circuits, including new weighted Monte Carlo spiking codes and massively parallel spiking neural assemblies. It shows that these architectures enable hybrids of data-driven and model-driven Monte Carlo inference that suffice for real-time probabilistic inference in probabilistic programs. It includes architectures for generating approximate samples for latent variables and for unbiased estimation of probability densities and importance weights. Crucially, these architectures can also be used recursively. They apply at the scale of individual latent variables, and to larger collections of latent variables, arising in both target models and in proposal distributions. This approach thus enables complex Monte Carlo inference architectures, with proposals defined by data-driven probabilistic programs, including artificial neural networks, whose outputs are reweighted and corrected via generative model-driven Monte Carlo inference. Expressiveness is illustrated via three examples: visual prey tracking by larval zebrafish; mental physics simulation by both humans and non-human primates; and recursive concept learning by human adults. Figure 1 shows three inference tasks, each previously studied in Bayesian cognitive science, for which our approach provides the first spiking neural implementations.

1. Spiking neural Monte Carlo

A. Weighted Monte Carlo spiking codes. A dynamic probabilistic program defines a joint density over a sequence $\mathbf{z}_{1:T}$ of latent states, and a sequence of observed data, $\mathbf{d}_{1:T}$:

$$P(\mathbf{z}_{1:T}, \mathbf{d}_{1:T}) = P(\mathbf{z}_1) \prod_{t=2}^{T} P(\mathbf{z}_t|\mathbf{z}_{t-1}) \prod_{t=1}^{T} P(\mathbf{d}_t|\mathbf{z}_t) \qquad [1]$$

Inference in a dynamic probabilistic program consists of estimating the conditional distribution $P(\mathbf{z}_{1:t}|\mathbf{d}_{1:t})$ for each t. This can be done sequentially, using the inference about $P(\mathbf{z}_{1:t-1}|\mathbf{d}_{1:t-1})$ to form inferences about $P(\mathbf{z}_{1:t}|\mathbf{d}_{1:t})$, using the following recursion:

$$P(\mathbf{z}_{1:T}|\mathbf{d}_{1:T}) = P(\mathbf{z}_{1:T-1}|\mathbf{d}_{1:T-1}) \times Q(\mathbf{z}_T; \mathbf{z}_{T-1}, \mathbf{d}_T) \frac{P(\mathbf{z}_T, \mathbf{d}_T|\mathbf{z}_{T-1})}{Q(\mathbf{z}_T; \mathbf{z}_{T-1}, \mathbf{d}_T) P(\mathbf{d}_T|\mathbf{d}_{T-1})} \qquad [2]$$

The sequential Monte Carlo algorithm implements this recursion for an approximation each distribution $P(\mathbf{z}_{1:t}|\mathbf{d}_{1:t})$ represented as a set $\{(\mathbf{z}^i_{1:t}, w^i_t)\}_{i=1}^N$ of "weighted particles" meant to approximate the distribution. At each time t, the existing particles of the form $\mathbf{z}^i_{1:t-1}$ are extended, and the weights are updated, according to

$$\mathbf{z}^i_t \sim Q(\cdot; \mathbf{d}_t, \mathbf{z}^i_{t-1}), \quad w^i_t = w^i_{t-1} \frac{P(\mathbf{z}^i_t|\mathbf{z}^i_{t-1}) P(\mathbf{d}_t|\mathbf{z}^i_t)}{Q(\mathbf{z}^i_t; \mathbf{d}_t, \mathbf{z}^i_{t-1})} \qquad [3]$$

B. Massively parallel micro-scale spiking assemblies and micro-circuits for individual latent variables. A conditional probability distribution $P(z|\mathrm{par}(z))$ is implemented using a collection of neural assemblies, one for each value i that z can take. The ith assembly has $a_P^{z=i}$ neurons in it, each spiking at rate $r_P^{z=i}$, given a particular value of $\mathrm{par}(z)$. The overall rate of the assembly is therefore $\lambda_P^{z=i} = a_P^{z=i} \times r_P^{z=i}$. The assemblies correctly implement $P(z|\mathrm{par}(z))$ if the rate of the each assembly is equal to the probability of the corresponding value of z, up to some proportionality constant γ_P:

$$\lambda_P^{z=i} = \gamma_P P(z = i|\mathrm{par}(z)) \qquad [4]$$

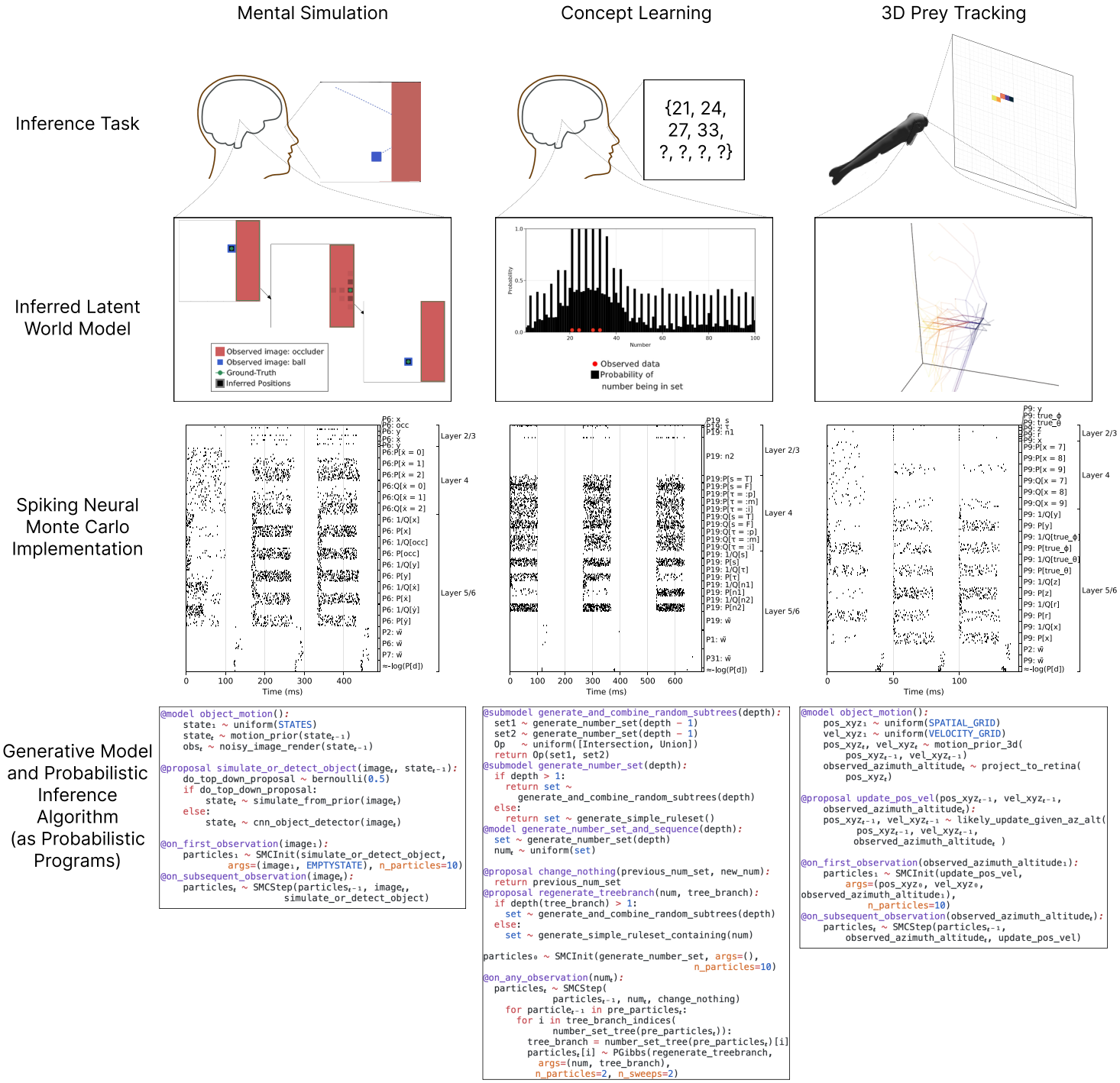


Fig. 1. Real-time spiking neural Monte Carlo models for diverse probabilistic inferences in perception and cognition. Tasks are primate mental simulation (left column), human concept learning (middle column), and 3D prey tracking (right column). Each task (top row) requires the model organism to infer latent world models (second row). This is achieved by a spiking neural Monte Carlo implementation (third row) of probabilistic programs (bottom row) that encode a generative model and a sequential Monte Carlo inference algorithm, implementing hybrids of data-driven and model-driven inference. Spike rasters show weighted Monte Carlo spiking representations distributed across model neurons from superficial (L2/3), middle (L4), and deeper layers of cortex (L5/6) including sparse codes, dense codes, and power oscillations in the gamma and theta bands. The theory in this paper shows how to automatically construct spiking models such as these that implement biologically realistic, massively parallel, real-time inference for a broad class of probabilistic programs.

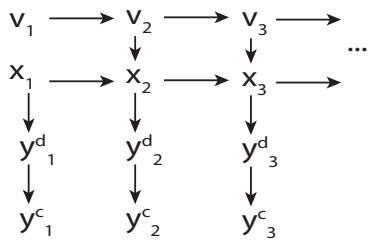
Likewise, a proposal distribution $Q(z; d)$ is implemented using a collection of assemblies, the ith having $a_Q^{z=i}$ neurons at rate $r_Q^{z=i}$ given d to achieve total rate $\lambda_Q^{z=i} = a_Q^{z=i} \times r_Q^{z=i}$, where

$$\lambda_Q^{z=i} = \gamma_Q Q(z = i; d) \qquad [5]$$

For $k > 0$ let $s_P^{i,k}$ be the time when the kth spike is emitted

```
@gen function initial_latent_model()
    x ~ uniform_discrete(1, 10)
    v ~ uniform_discrete(-3, 3)
end
@gen function step_latent_model(
        x_prev, v_prev
    )
    v ~ discretized_gaussian(v_prev, 0.2)
    x ~ exactly(x_prev + v)
end
@gen function obs_model(x, v)
    y_disc ~ discretized_gaussian(x, 0.5)
    y_cont ~ gaussian(y_disc, 0.4)
end
```

(a) Probabilistic generative model for 1D tracking

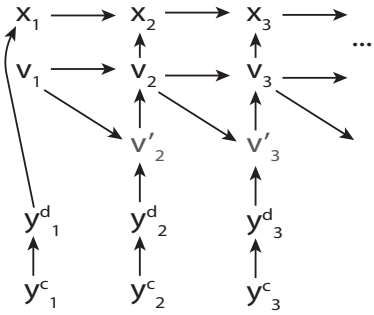


(b) Graphical model corresponding to (a).

```
# activation of each tuning-curve, given y_cont
tuning_curves(y_cont) = normalize([
    pdf(gaussian, (mean, 0.4), y_cont)
    for mean in 1:10
])
@gen function initial_proposal(y_cont)
    y_disc ~ cat(tuning_curves(y_cont))
    v ~ uniform_discrete(-3, 3)
    x ~ discretized_gaussian(y_disc, 0.5)
end

@gen function step_proposal(x_prev, v_prev, y)
    y_disc ~ cat(tuning_curves(y_cont))
    v' = y_disc - x_prev # apparent velocity
    v ~ discretized_gaussian((v_prev + v')/2, 1)
    x ~ exactly(x_prev + v)
end
```

(c) Data-driven inference proposals



(d) Graphical model corresponding to (c).

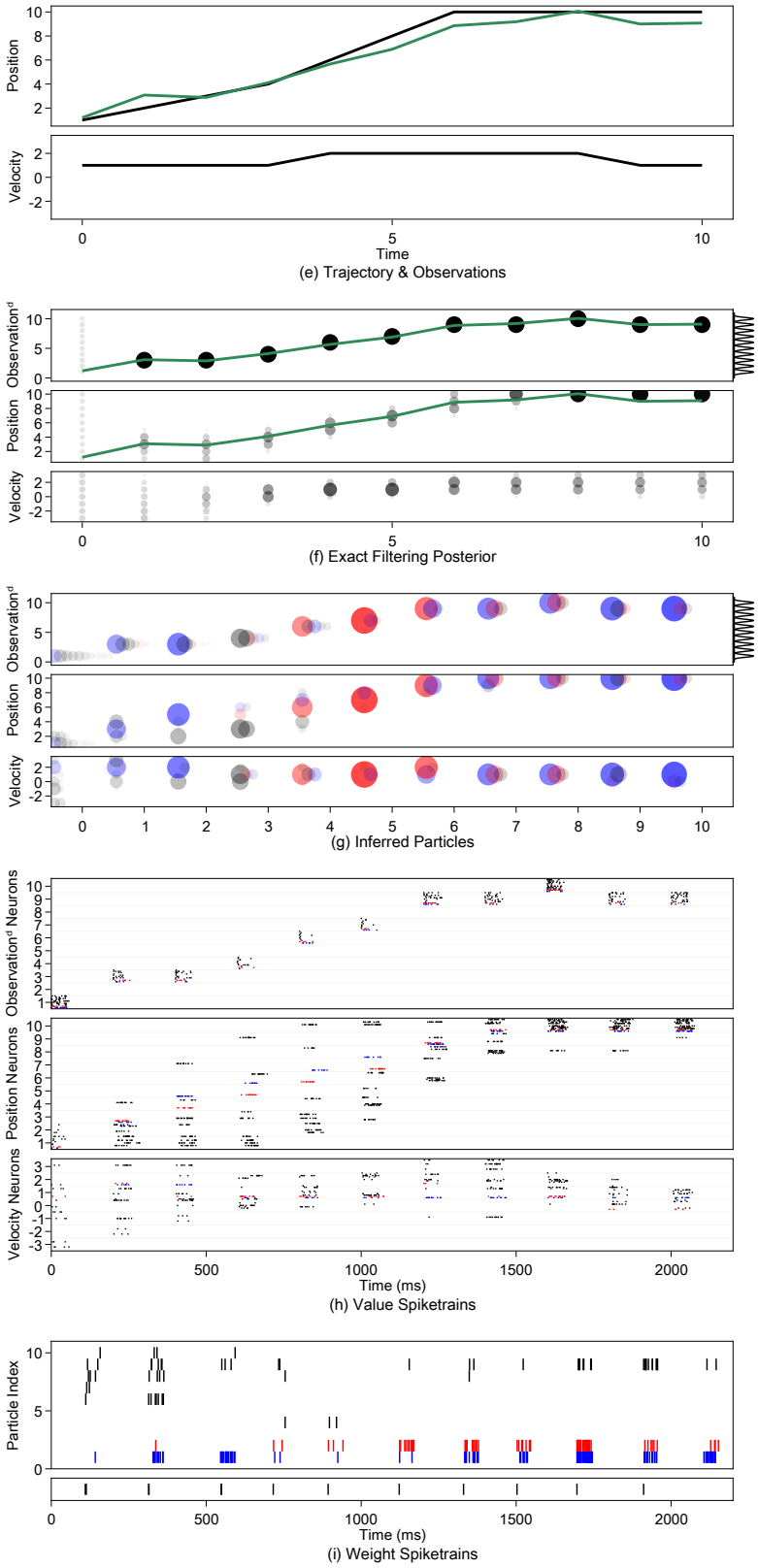


Fig. 2. A weighted Monte Carlo spiking code for a 1 dimensional visual prey tracking problem. (a) shows the target probabilistic generative model, implemented in Gen, and (b) shows its top-down dependency structure as a directed graphical model. (c) shows bottom-up, data-driven inference proposals, along with (d) their dependence structure. (e) shows the inference problem being solved: tracking a prey moving in 1 dimension. The ground truth prey trajectory is shown in in black, and the observed position data is shown in green. (f) shows an exact Bayesian filtering solution to this problem. Note that observed position is discretized via Gaussian tuning curves before conditioning the Bayes filter. (g) shows a particle filtering approximation, with particle importance weights encoded by circle size. (h) shows sparse codes for sampled values representing the particles from (g), and (i) shows dense codes for estimated importance weights from (g).

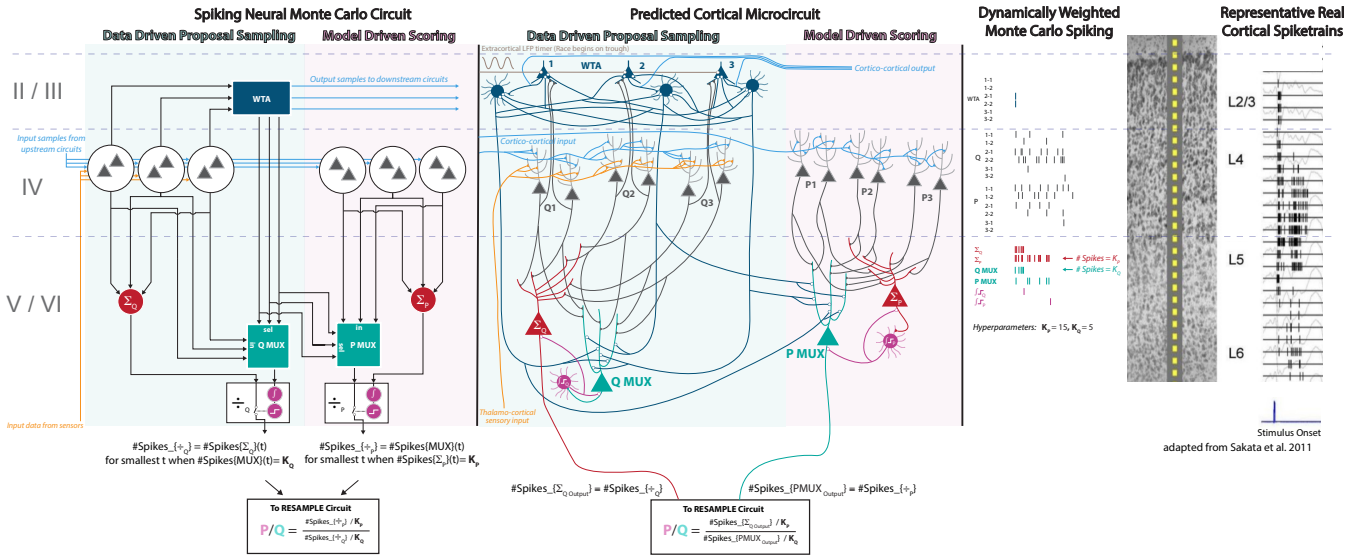


Fig. 3. Micro-scale spiking assemblies and spiking neural Monte Carlo micro-circuits predict connectivity, coding, and dynamics of real cortical micro-circuits. From left to right, this figure shows the architecture for proposing and scoring a single latent variable; the predicted connectivity and synaptic characteristics of a real cortical micro-circuit implementing this design; representative spiking at each layer; and real cortical spiking from a depth electrode recording. Each spiking assembly represents a possible value for a random variable. The sampled proposal value is generated via a winner-take all race, which feeds into MUXs to generate spike counts. These spike counts yield provably unbiased estimates of proposal and target probabilites, and can be combined into importance weights. This architecture turns out to predict multiple features of the synaptic physiology, connectivity, and spiking dynamics of cortical micro-circuits.

from the P-assembly corresponding to outcome $z = i$, and let $s_P^{i,0} = 0$. We model the overall assembly as a Poisson Process, meaning that for each i and each k, the time between the k and $k+1$th spike is exponentially distributed:

$$s_P^{i,k} - s_P^{i,k-1} \overset{\text{i.i.d.}}{\sim} \text{Exponential}(\lambda_P^{z=i}) \qquad [6]$$

$S_P^0 = \{s_P^{i,k}\}_{i,k}$ is the set of the times at which any P assembly spikes. Let S_P^k be the set of all spike-times except the first k, and let t_P^k be the time at which the kth spike is emitted from any assembly:

$$t_P^k = \inf S_P^{k-1}, \qquad S_P^k = S_P^{k-1} \setminus \{t_P^k\} \qquad [7]$$

Let z_P^k be the value i of z corresponding to the assembly which emitted the kth spike among all the spikes:

$$z_P^l = i : t_P^k \in \{s_P^{i,k}\}_k \qquad [8]$$

Likewise we let $s_Q^{i,k}$ denote the time of the kth spike from the ith Q-assembly, and we define S_Q^k, t_Q^k, and z_Q^k analogously:

$$\forall k > 0, s_Q^{i,k} - s_Q^{i,k-1} \overset{\text{i.i.d.}}{\sim} \text{Exponential}(\lambda_Q^{z=i}) \qquad [9]$$

$$\forall k > 0, \quad t_Q^k = \inf S_Q^{k-1}, \quad S_Q^k = S_Q^{k-1} \setminus \{t_Q^k\} \qquad [10]$$

$$\forall k > 0, z_Q^k = i : t_Q^k \in \{s_Q^{i,k}\}_k \qquad [11]$$

where $S_Q^0 = \{s_Q^{i,k}\}_{i,k}$ and $s_Q^{i,0} := 0$.

It turns out that since the assemblies are Poisson Processes (Eqns. 6, 9) with appropriately set rates (Eqns. 4, 5), the values z_P^k and z_Q^k are fair samples:

$$z_P^k \overset{\text{i.i.d.}}{\sim} P(z|\text{par}(z)), \quad z_Q^k \overset{\text{i.i.d.}}{\sim} Q(z;d) \qquad [12]$$

From this it is evident how to draw a sample from the Q proposal distribution: simply use one of the z_Q^k values! In our proposed circuits, the identity $i^* = z_Q^1$ of the first assembly to spike is selected by a Winner-Take-All circuit to be used as the sampled value, and output in the sparse code.

Since each spike index is a fair sample from P or Q, we can obtain approximations of $P(z = i^*|\text{par}(z))$ and $Q(z = i^*; d)$ using simple Monte Carlo estimates. To do this for a P-score, we use a circuit that considers what fraction of the first c_P spikes came from the i^*th assembly. Let $N_P^{z=i}$ be the number of spikes from the ith assembly, out of the first c_P spikes

$$N_P^{z=i} = \sum_{k=1}^{c_P} 1_{z_P^k = i}, \qquad [13]$$

and let N_P^z denote the count for the sampled value, $N_P^z = N_P^{z=i^*}$. Let$\hat{p}$ be the fraction of the first c_P spikes to come from the i^*th assembly:

$$\hat{p} = \frac{1}{c_P} N_P^z \qquad [14]$$

Then $\hat{p}$ is an unbiased estimate of $p := P(z = i^*|\text{par}(z))$: $\mathbb{E}[\hat{p}] = p$.

For the Q distribution we obtain a probability-value estimate slightly differently, because (1) we wish to obtain an unbiased estimate of $q^{-1} = 1/Q(z = i^*; d)$, rather than of q, and (2) the first spike z_P^1 equals the sampled value i^* (since this is how the circuit samples i^*) and therefore this first spike must be ignored. Due to (1), rather than waiting for a fixed number of spikes to occur from any assembly, the $1/Q$-scoring circuit waits until c_Q spikes occur *in the* i^*th *assembly*. Let $N_Q^{z=i}$ be the number of spikes from the ith assembly by the time that the $c_Q + 1$th spike is emitted by the i^*th assembly,

$$N_Q^{z=i} = \sum_k 1_{s^{i,k} \leq s^{i^*, c_Q+1}}, \qquad [15]$$

and let N_Q^z be the number of spikes from *all* the assemblies in this time, excluding the first spike: $N_Q^z = \sum_i N_Q^{z=i} - 1$.

Then let $\hat{q}^{-1}$ be the ratio of the total number of spikes in any assembly to the number from the i^*th assembly in this time:

$$\hat{q}^{-1} = \frac{1}{c_Q} N_Q^z \qquad [16]$$

Then $\hat{q}^{-1}$ is an unbiased estimate of q: $\mathbb{E}[\hat{q}^{-1}] = q$.

Let $\tau_{Q,z}^{\text{sample}}$ be the amount of time to generate a sample of z from Q, and let $\tau_{Q,z}^{\text{score}}$ be the amount of time needed to Q-score the sample after it has been drawn,

$$\tau_{Q,z}^{\text{sample}} = t_Q^1, \quad \tau_{Q,z}^{\text{score}} = s_Q^{i^*, c_Q} \qquad [17]$$

so

$$\tau_{Q,z}^{\text{sample}} \sim \text{Exponential}(\sum_i \lambda_Q^i),$$

$$\tau_{Q,z}^{\text{score}} - \tau_{Q,z}^{\text{sample}} \sim \text{Erlang}(c_Q + 1, \lambda_Q^{i^*})$$

The amount of time $\tau_{P,z}^{\text{score}}$ needed to P-score z is defined analogously.

C. Massively parallel meso-scale spiking networks for high--dimensional probabilistic programs. For simplicity of presentation we assume that the latent state and data vectors $\mathbf{z}_t$ and $\mathbf{d}_t$ have fixed length regardless of t, and we use the indices $\{1, \ldots, |\mathbf{z}_t|\}$ to refer to the variables in $\mathbf{z}_t$ and the indices $\{|\mathbf{z}_t| + 1, \ldots, |\mathbf{z}_t| + |\mathbf{d}_t|\}$ to refer to the variables in $\mathbf{d}_t$.

Since the step model is a probabilistic program, the density $P(\mathbf{z}_t|\mathbf{z}_{t-1})$ decomposes into the product

$$P(\mathbf{z}_t|\mathbf{z}_{t-1}) = \prod_{i=1}^{|\mathbf{z}_t|} P(\mathbf{z}_t^i | \{\mathbf{z}_t^j\}_{j \in \text{par}_P^t(i)}, \{\mathbf{z}_{t-1}^j\}_{j \in \text{par}_P^{t-1}(j)}) \qquad [18]$$

where $\text{par}_P^t(i)$ and par_P^{t-1} are the indices of the parent variables of variable i in $\mathbf{z}_t$ and $\mathbf{z}_{t-1}$ respectively.

Likewise, the Q proposal probabilistic program decomposes into a product. Since values sampled from Q proposal distributions must be sampled in the topological order induced by the probabilistic program, we emphasize that the variables proposed by Q are organized into Depth_Q layers L_Q^1 through $L_Q^{\text{Depth}_Q}$, which together form a partition of the set of variable indices $\{1, \ldots, |\mathbf{z}_t|\}$. $Q(\mathbf{z}_t; \mathbf{z}_{t-1}, \mathbf{d}_t)$ decomposes into

$$Q(\mathbf{z}_t; \mathbf{z}_{t-1}, \mathbf{d}_t) = \prod_{k=1}^{\text{Depth}_Q} \prod_{i \in L_Q^k} Q(\mathbf{z}_t^i; \{\mathbf{z}_t^j\}_{j \in \text{par}_Q^t(i)}, \{\mathbf{z}_{t-1}^j\}_{j \in \text{par}_Q^{t-1}(i)}, \{\mathbf{d}_t^j\}_{j \in \text{par}_Q^d(i)}) \qquad [19]$$

par_Q^t, par_Q^{t-1}, and par_P^d are the indices of the parent variables of variable i in $\mathbf{z}_t$, $\mathbf{z}_{t-1}$, and $\mathbf{d}_t$ respectively:

Let $\tau_{Q,L_Q^k}^{\text{sample}}$ be the maximum time needed to sample any variable in L_Q^k, let τ_Q^{score} denote the maximum time needed to score any variable in $\mathbf{z}_t$, and let τ_P^{score} denote the maximum time needed to score any variable in $\mathbf{z}_t$ or $\mathbf{d}_t$.

$$\tau_{Q,L_Q^k}^{\text{sample}} = \max_{i \in L_Q^k} \tau_{Q,i}^{\text{sample}} \qquad [20]$$

$$\tau_Q^{\text{score}} = \max_{i=1}^{|\mathbf{z}_t|} \tau_{Q,i}^{\text{score}}, \quad \tau_P^{\text{score}} = \max_{i=1}^{|\mathbf{z}_t + \mathbf{d}_t|} \tau_{P,i}^{\text{score}} \qquad [21]$$

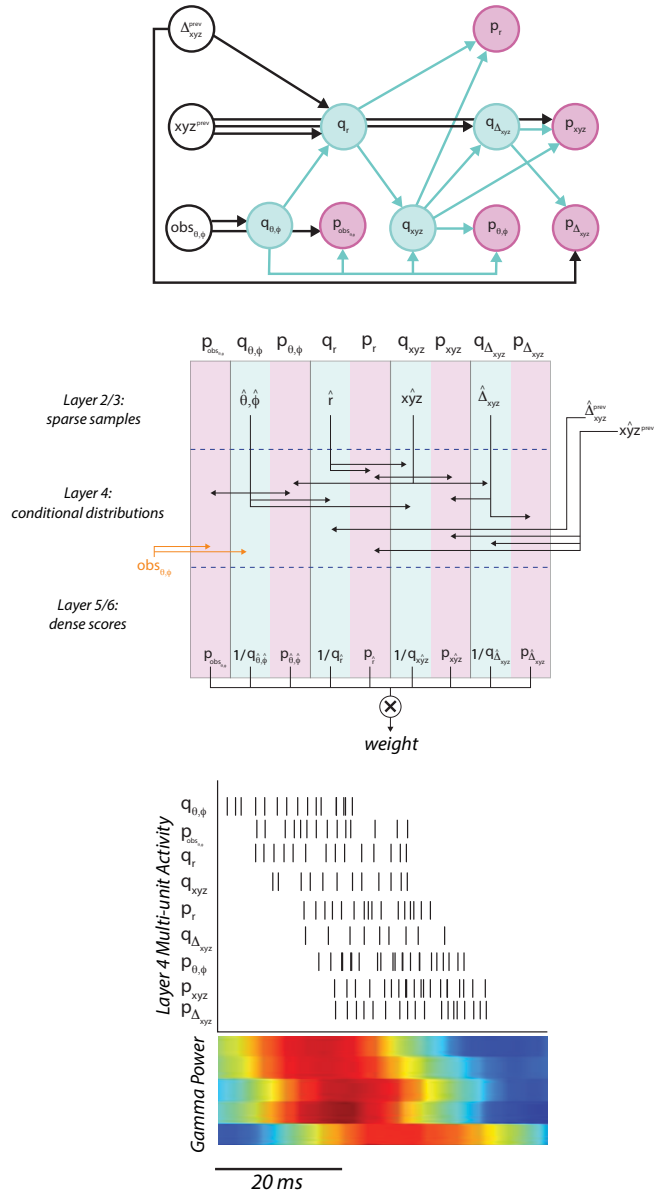


Fig. 4. Meso-scale spiking monte Carlo networks for sampling and scoring multiple variables. (top) the dependence structure of one time slice in the 3D prey tracking model, showing how proposed values and scoring can be interleaved and parallelized. (middle) Each variable's data-driven proposal sampler and model-based scoring circuit is located in its own micro-circuit. The graph structure above is implemented via inter-micro-circuit connections at the appropriate layers. (bottom) This architecture predicts traveling spiking cascades, spreading across dependent columns at the speed of gamma oscillations, that have been confirmed in multiple model organisms. Note that latency is low — only long enough to get a single sampled value from the layer II/III WTAs — because all scoring can be done via massive parallelism. Also note that all variables in the target model can be scored at the same time, regardless of its size.

	Latent Variables	Observed Variables	Size of Spiking Neural Representation: Weighted Monte Carlo (this paper)	Size of Spiking Neural Representation: ENS Codes, Standard PPCs
1D object tracking	$\{x_t, \dot{x}_t\}_t$	$\{d_t^x\}_t$	Sparse: 27 Dense: 5	Dense: 140
2D object tracking	$\{x_t, y_t, \dot{x}_t, \dot{y}_t\}_t$	$\{d_t^x, d_t^y\}_t$	Sparse: 30 Dense: 10	Dense: 2500
Mental Physics Simulation	$\{x_t, y_t, \dot{x}_t, \dot{y}_t, o_t\}_t$	$\{((d_t^{(i,j)})_{i=1}^{10})_{j=1}^{10})\}_t$	Sparse: 38 Dense: 110	Dense: 2500
3D object tracking from 2D observations	$\{x_t, y_t, z_t, \dot{x}_t, \dot{y}_t, \dot{z}_t, r_t, \phi_t, \theta_t\}_t$	$\{d_t^\phi, d_t^\theta\}_t$	Sparse: 160 Dense: 20	Dense: 23,180,062,500
Recursive Concept Learning (Sizes are for $D = 2, M = 10$)	$\bigcup_{h=1}^{D} \bigcup_{b=1}^{2^{h-1}} \{s^{(h,b)}, \tau_r^{(h,b)}, \tau_c^{(h,b)}, n_1^{(h,b)}, n_2^{(h,b)}\}$	$\{d_t\}_{t=1}^{M}$	Sparse: 180 Dense: 40	Dense: 5.92704×10^{11}

Table 1. Weighted Monte Carlo spiking requires exponentially fewer neurons than standard probabilistic population codes and ENS spiking codes. For low-dimensional probabilistic programs that only make a small number of latent choices, the difference can be modest in aboslute terms. As the number of latent variables in the probabilistic program grows, the cost of the neural representation for previously proposed schemes grows exponentially, rendering them impractical for the majority of perceptual and cognitive inferences.

Then the overall latency needed at each timestep of SMC to sample $\mathbf{z}_t$ and estimate the importance weight update w_t/w_{t-1}, τ, is bounded by

$$\tau \leq \sum_{k=1}^{\text{Depth}_Q} \tau^{\text{sample}}_{Q,L_Q^k} + \tau^{\text{score}}_Q + \tau^{\text{score}}_P \qquad [22]$$

Observe that while the sampling time grows linearly in the *depth* of Q,the scoring time does not depend on the depth of Q or P, and given a fixed Q-depth, adding more variables to the model also does not increase latency. That is, arbitrarily large models P can be used to provide top-down feedback without linearly* increasing the latency of the circuit, since all the variables in the model can be scored in parallel.

D. Massively parallel macro-scale spiking circuits for real–time sequential Monte Carlo. The cortical columns for particle i corresponding to the Q sampler, Q scorer, and P scorer for each variable in $\mathbf{z}_t^i$, and the columns for P-scoring each variable in $\mathbf{d}_t$, output a vector $\mathbf{z}_t^i$ of sampled variable values represented in the sparse code, and output two collections of spike counts, $\{N_P^{i,j}\}_{j=1}^{|\mathbf{z}_t|+|\mathbf{d}_t|}$ and $\{N_Q^{i,j}\}_{j=1}^{|\mathbf{z}_t|}$ (these are the counts N_P^z and N_Q^z used in Eqns. 14, and 16 where z is the jth variable in the ith particle). These spike counts encode the set of probability estimates $\{\hat{p}_{i,j}\}_{j=1}^{|\mathbf{z}_t|+|\mathbf{d}_t|}$ and $\{(\hat{q}_{i,j})^{-1}\}_{j=1}^{|\mathbf{z}_t|}$ via the relations

$$\hat{p}_{i,j} = N_P^{i,j}/c_P, \quad \hat{q}_{i,j}^{-1} = N_Q^{i,j}/c_Q \qquad [23]$$

While each of these score terms can be approximated reasonably well using a spike count from a single neuron or assembly that is proportional to the value $\hat{p}$ or $\hat{q}$ by a fixed constant c_P or c_Q, the overall importance weight estimate $\hat{w}_i = \prod_{j=1}^{|\mathbf{z}_t|+|\mathbf{d}_t|} \hat{p}_{i,j} \prod_{j=1}^{|\mathbf{z}_t|} \hat{q}_{i,j}^{-1}$ can have enormous dynamic range, and so cannot be represented with a spike count proportional to the value with a fixed constant of proportionality.† To alleviate this issue, when multiplying $\hat{p}$ and $\hat{q}$ terms to compute the importance weight estimates $\hat{w}_i$, we use a circuit that dynamically chooses the constant of proportionality used in the spiking representation of $\hat{w}$, and represents this value using a count $c_{\log}$ on a logarithmic scale (so that large ranges of values can be represented). The constant of proportionality is chosen to be the same for each particle, so that the different importance weights $(\hat{w}_i)_i$ can be compared directly to one another. This is implemented using the `MultAutonorm` circuit, which has $N+1$ output assemblies, one to output the value $c_{\log}$, and N to output a spiking rate λ_i for each particle:

$$(c_{\log}, (\lambda_i)_{i=1}^N) \sim \texttt{MultAutonorm}((\{N_P^{i,j}\}_{j=1}^{|\mathbf{z}_t|+|\mathbf{d}_t|}, \{N_Q^{i,j}{}_{i,j}\}_{j=1}^{|\mathbf{z}_t|})_{i=1}^N) \qquad [24]$$

These outputs convey the importance weight values in a representation called *neural floating point*, where one line is on a logarithmic scale and the other is on a direct scale (similarly to the way floating-point numbers are represented in digital computers):

$$\hat{w}^i = b^{c_{\log}} \lambda_i \qquad [25]$$

where b is the base of the logarithm. The spike rates λ_i can be read-out into spike counts and sent to other parts of the circuit as estimates of $\hat{w}^i/b^{c_{\log}}$, or these assemblies can themselves be used as a sampler (using the same WTA circuit used in the Q samplers for sampling each variable) to choose a particle with probability proportional to its importance weight. This is the key operation needed to perform resampling. Crucially, the `MultAutonorm` will choose a $c_{\log}$ value so that the sum $\sum_i \lambda_i$ of the output spiking rates has a small dynamic range (roughly, a range of 10-100 Hz), so that it is possible to read-out spikes from some of the assemblies relatively quickly, and none of the assemblies are saturated. For the details of this circuit, see Appendix **??**.

* In our model of assemblies as Poisson-Processes, where the $\tau^{\text{score}}_{P,i}$ values are Erlang-distributed, the expected latency will *slightly* increase as more variables are added to the model, because this increases the probability that one of the variables happens to take unusually long to score. In other models of neural scoring assemblies – e.g. as units which either spike or do not during each fixed time window (like (47)) – increasing the number of variables would not at all increase the expected scoring latency.

† For example, if a neuron's maximum rate is 100Hz and the maximum importance weight value is 1, we need 100Hz neurons to correspond to $\hat{w} = 1$. But then in cases where $\hat{w} = 10^{-10}$, we'd need to read-out a spike count from a neuron with rate 10^{-8} Hz. Reading this out at any reasonable precision would require waiting for years, or using an assembly with millions of neurons!

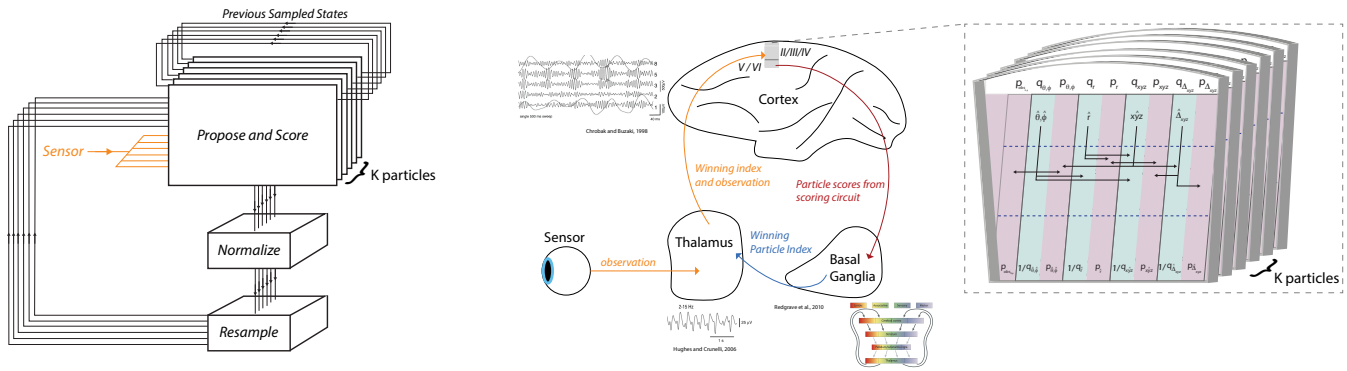


Fig. 6. Macro-scale, massively parallel spiking architectures for resampling that predict large-scale features of brain connectivity and dynamics. (left) A massively parallel resampling circuit that leverages multiple particles to provide more stable inferences for dynamic models. (middle) This architecture predicts the existence of brain structures that couple microcircuits across multiple brain regions to calculate particle weights, and that return the indices of selected particles back to the same source microcircuits. (right) Each particle has an independently evolving copy of the multivariate model shown previously. (52) and others have found this pattern of connectivity in the cortico-subcortical-thalamic loop. One hypothesis is that normalization and resampling happen in subcortical and thalamic regions. This architecture also predicts layering of oscillatory cycles for sampling and scoring within each particle with larger oscillations for sampling and scoring across particles. (53) and others have found that gamma oscillations are embedded within theta oscillations, and (54) and others have suggested that the thalamus (which can read out when resampling has finished) generates the "clock signals" that drive theta oscillations.

Given weighted particles $(\mathbf{z}_t^i, w_t^i)_{i=1}^N$, resampling outputs a new collection $(\mathbf{z}_t^{[i]}, w_t^{[i]})_{i=1}^N$, where

$$\forall i, \mathbf{z}_t^{[i]} = \mathbf{z}_t^{a_t^i} \text{ where } a_t^i \sim \text{Cat}([\tilde{w}_t^1, \ldots, \tilde{w}_t^N]) \qquad [26]$$

and $\forall i, w_t^{[i]} = 1$.

2. Scaling via hybrids of data-driven and model-driven inference

An L-layer artificial neural network with K_l neurons in layer l computes the activity vector $\mathbf{x}^l$ at each layer l according to

$$\mathbf{x}_n^l = f^l(\mathbf{x}^{l-1})_n = \sigma(w_{0,n}^l + \sum_{m=1}^{K_{l-1}} w_{m,n}^l \mathbf{x}_m^{l-1}) \qquad [27]$$

The overall neural network operates on data matrix $[\mathbf{d}_{i,j}]_{[i,j]}$ and implements the computation

$$f([\mathbf{d}_{i,j}]_{i,j}) = \mathbf{x}^L = f^L(\cdots f^2(f^1([\mathbf{d}_{i,j}]_{i,j}))) \qquad [28]$$

The output vector defines a probability distribution over variable z by

$$P(z = i|[\mathbf{d}_{i,j}]_{i,j}) = \frac{\mathbf{x}_i^L}{\sum_{n=1}^{K_L} \mathbf{x}_n^L} \qquad [29]$$

We can implement the neural network by using vectors of spiking rates, λ^l, to represent the activity vectors $\mathbf{z}^l$, using a network of neurons arranged so that the rate of the nth neuron in layer l to

$$\lambda_n^l = \sigma(w_{0,n}^l + \sum_{m=1}^{K_{l-1}} w_{m,n}^i \lambda_m^{l-1}) \qquad [30]$$

Sampling from the last layer, of this network, λ^L, can be performed by treating each neuron in the output layer as a neural assembly, and using the proposed circuits for sampling or scoring from neural assemblies.

The probability of a pixel given the deterministic rendering of that pixel from the latent state, $P(d_{i,j}|r_{i,j})$, is computed by marginalizing over two latent variables, u_1 and u_2. It turns out that this can be approximated by *pseudo-marginalizing* over these variables using Spiking Neural Monte Carlo to propose values for u_1 and u_2, and evaluate the known density $P(d_{i,j}, u_1, u_2|r_{i,j})$ given those values:

$$P(d_{i,j}|r_{i,j}) = \sum_{u_1,u_1} P(d_{i,j}, u_1, u_2|r_{i,j})$$
$$\approx \frac{P(d_{i,j}, u_1, u_2|r_{i,j})}{Q(u_1, u_2; d_{i,j}, r_{i,j})} \text{ where } u_1, u_2 \sim Q(\cdot; d_{i,j}, r_{i,j}) \qquad [31]$$

After resampling, resampled particle $\mathbf{z}_t^{[i]}$ may be run through a sequence of MCMC kernels T_i to produce a rejuvenated sample $\mathbf{z}_t^{(i)}$

$$\mathbf{z}_t^{(i)} \sim (T_k \circ T_{k-1} \circ \cdots \circ T_1)(\cdot; \mathbf{z}_t^{[i]}) \qquad [32]$$

Particle Gibbs defines an MCMC transition kernel $\text{PG}_{Q,\mathbf{d}_t,\mathbf{z}_{t-1}}$ which depends on a proposal distribution Q, the data $\mathbf{d}_t$, and the previous latent state $\mathbf{z}_{t-1}$ that the particle to rejuvenate, $\mathbf{z}_t^{[i]}$, was generated in connection with (that is, $\mathbf{z}_{t-1}^{[a_t^i]}$).

To sample a new state from $\mathbf{z}_t^* \sim \text{PG}_{Q,\mathbf{d}_t,\mathbf{z}_{t-1}}(\cdot; \mathbf{z}_t)$, the following steps are run:

$$\mathbf{z}_t' \sim Q(\cdot; \mathbf{z}_{t-1}, \mathbf{z}_t, \mathbf{d}_t) \qquad [33]$$

$$\alpha \sim \text{Bernoulli}\left(\frac{\frac{P(\mathbf{z}_t', \mathbf{d}_t|\mathbf{z}_{t-1})}{Q(\mathbf{z}_t'; \mathbf{z}_{t-1}, \mathbf{z}_t, \mathbf{d}_t)}}{\frac{P(\mathbf{z}_t', \mathbf{d}_t|\mathbf{z}_{t-1})}{Q(\mathbf{z}_t'; \mathbf{z}_{t-1}, \mathbf{z}_t, \mathbf{d}_t)} + \frac{P(\mathbf{z}_t, \mathbf{d}_t|\mathbf{z}_{t-1})}{Q(\mathbf{z}_t; \mathbf{z}_{t-1}, \mathbf{z}_t', \mathbf{d}_t)}}\right) \qquad [34]$$

$$\mathbf{z}_t^* = 1_\alpha \mathbf{z}_t' + 1_{\neg\alpha} \mathbf{z}_t \qquad [35]$$

3. Discussion

How can this theory be tested more thoroughly against empirical data? One approach is to build and test larger-scale spiking Monte Carlo circuit models of brain systems for 3D scene perception and navigation. These can be grounded in structured latent world models that integrate probabilistic

Characteristics of fundamental building blocks of spiking neural Monte Carlo	Experimental evidence for their existence in biological neural systems
Samples are represented via sparse codes from WTAs, but scores are represented via dense codes from MUXes and assemblies	Sparse & dense codes coexist in multiple brain regions (57)
Samples are generated via first-to-spike races between Poisson processes	EPSCs and IPSCs in all neural systems studied to date follow an exponentially-distributed spacing rule, i.e. the number of spikes in a given time window follows a Poisson process (58, 59). Increased synaptic input yields a change in Poisson rate, i.e. probability of race victory scales directly with input.
Winner-take-all samplers rely on fast inhibition.	Ephaptic coupling enables inhibition at the speed of electrical propagation (60, 61)
Scoring units rapidly and accurately count spikes from MUXes and assemblies.	NMDAR plateau potentials are a recently discovered non-decaying synaptic current (50ms) that can stack linearly with other arriving potentials, providing a mechanism for short timescale counting of presynaptic spikes. (62, 63)

occupancy grids, 3D scene graphs (18), and hierarchical object models that adjust resolution based on perceptual uncertainty (77). In both non-human primates and rodents, these models could be compared to fine-grained neural data using relatively well-established techniques (78), and also simultaneously compared to fine-grained behavioral measurements of reaction time and accuracy. It seems appealing to use spiking neural Monte Carlo to integrate empirical constraints that neither Bayesian cognitive models nor artificial neural networks can precisely account for, such as the number of neurons and connectivity of the dorsal and ventral streams, and quantitative latency/accuracy tradeoffs that are observed both neurally and behaviorally. It remains to be seen whether quantitative predictions can be made precisely enough to motivate direct comparison of interventions on model networks to interventions on biological neurons.

Another approach, grounded in cognitive neuroscience, is to seek spatiotemporally coarser behavioral and neural correlates that are easier to measure via neuroimaging techniques. For example, the timecourse of traveling gamma waves, aligned with connectomic data, constrains the dependence structure of data-driven proposals for 3D perception via inverse graphics (22, 79), and also the dependence structure of top-down generative models. Quantitative similarities between weighted Monte Carlo spiking activity could also potentially be compared to behavioral similarity measures and to similarity between stimulus-induced BOLD activity.

Complementary tests can be obtained via smaller model organisms and also via in vitro studies, leveraging their vastly greater levels of observability and control. For example, the spiking model of 3D visual prey tracking from this paper already gives a more detailed causal account of prey capture than previous phenomenological models grounded in Bayesian cognitive science (80), and suggests an approach to depth estimation that could explain recent data on 3D barrier avoidance (81). It also seems appealing to implement spiking neural Monte Carlo circuits using detailed biophysical simulators (82, 83), and to compare implementations against quantitative data from in vitro experiments.

A. Scaling to richer forms of cognition and learning. Bayesian cognitive scientists can directly apply the theory in this paper to make more fine-grained resource-rational models of causal reasoning. For example, spiking neural circuits for model-driven particle Gibbs MCMC could potentially be fit to population-level inference latency and accuracy. Unlike standard resource-rational models, spiking neural Monte Carlo model fits could incorporate quantitative assumptions about the number of neurons and the level of parallelism that is recruited by the thought process. Spiking neural Monte Carlo can also be used to implement richer models of thinking processes that leverage inference-based value-of-information estimators (84, 85).

Real-time perceptual learning, real-time inference over dynamic data structures (13), and structure learning of probabilistic programs (86) all present additional challenges. For example, although probabilistic programs can learn models of novel objects from just a handful of images (18), they have not yet been shown to simultaneously learn the structure of new generative models for objects and the structure of new efficient data-driven proposals for recognizing those objects. This paper shows how the training data for self-supervised learning of data-driven neural network proposals could be generated, and how probabilistic losses could be rapidly estimated. It does not reveal how to scalably estimate gradients, even for shallow models. It is unclear if biologically realistic deep learning implementations can be developed, or if it will be more fruitful to pursue alternatives based on shallow learning and geometric modeling (87, 88) or online synthesis of provably near-optimal data-driven proposals (89). It is also unclear which truncated representations of latent data structures (such as 3D scene graphs, the plans of other agents, syntactic parse trees and logical representations of grounded natural language semantics,and even symbolic probabilistic program source code representing learned concepts) will lead to practical spiking neural Monte Carlo circuits for real-time inference.

B. Risk-sensitive control, action selection, and planning via inference. Brain computation requires risk-sensitive action selection, not just uncertain inference about world structure. The theory of spiking neural Monte Carlo can be applied to yield neurally mappable architectures for risk-sensitive model-based predictive sensorimotor control, action selection, and planning. This can be achieved via well-known reductions of those problems to probabilistic inference, such as (90–92).

Characteristics of micro-circuits for single-variable importance sampling	Characteristics of biological cortical micro-circuits
WTA units fire only sparsely, at the beginning of each sample/score cycle, to enforce a single race winner.	Layer II/III fires the most sparsely of the cortical layers. (64, 65)
Assembly neurons receive parents' sampled values from parents' WTA samplers, as well as sensory observations.	Layer IV receives intracortical input from Layer II/III (WTAs) of other microcolumns, plus sensory input from the thalamus (66)
WTA neurons control which assembly's spikes pass through the MUX	Layer II/III sends inhibitory projections to Layer V dendrites (67)
Multiplexer units collect spikes from all assemblies, but only output spikes from the assembly chosen by the WTA	Dendritic segmentation of input channels has recently been discovered, providing a biophysically realistic implementation mechanism (68, 69)

Characteristics of meso-scale multivariate importance sampling	Meso-scale characteristics of biological neural systems
Multivariate importance sampling requires latency sufficient for proposal cascades and synchronized scoring, yielding traveling cascades of layer 4 spiking across micro-circuits, at the latency needed for individual samples	Gamma-band (30-100Hz) oscillations and traveling waves (70) are widely observed
Latency τ for sampling and scoring depends on data and parent values, and is thus variable across brain states and regions	Gamma-band oscillations are predicted to have variable frequency, as is widely observed
Assemblies representing more probable values will spike earlier relative to traveling cascades	Phase precession of spiking with respect to gamma oscillations has been observed (71)
Global posterior probabilities can only be read via normalized weights, not directly via spike rates, thus "beliefs" are only implicit and challenging to extract from weighted Monte Carlo spiking	Direct mappings of posterior probabilities and environmental probabilities onto firing rates are not yet strongly supported by empirical evidence (12)

Characteristics of macro-scale sequential Monte Carlo	Macro-scale connectivity and dynamics of biological neural systems
Resampling draws on weights that span multiple microcircuits and returns new particle indices (for new sources of parent variables) back to source microcircuits	There is a cortically-stratified cortico-subcortical-thalamic loop that sends information from lower cortical layers (where SNMC predicts weights are stored) to the basal ganglia, and then back through the thalamus to source cortical layers (52)
Resampling of particles containing multiple variables takes more time than sampling single variables, and is synchronized across particles.	There exist larger-scale oscillations (e.g. alpha, theta) that embed gamma within them (53, 72), with rhythms generated at/near putative thalamic source of resampling (54)
Resampled particles can be used to make high-quality multivariate proposals (25, 73). Higher-quality proposals (i.e. closer to the local posterior distribution, generating higher posterior probability values for latent variables) will have higher weights, and can therefore be proposed more quickly (relative to the cycle at which particles are resampled).	Traveling waves are observed for slower oscillations, e.g. alpha and theta (74). Also, phase precession is observed relative to slower theta rhythms, e.g. for hippocampal place cells, which spike earlier relative to theta oscillations for place cells representing more probable places (75) and of activity in other regions such as the mPFC (76)

Structured, "program-like" policies for selecting actions can also be inferred using Monte Carlo inference in probabilistic programs (93). However, despite the potential engineering appeal of these approaches, their potential reverse-engineering value has yet to be evaluated. It also is possible that the brain leverages specialized neural mechanisms for optimizing risk-sensitive action selection beyond what can easily be achieved via Monte Carlo inference.

C. Fundamental limits on scale and efficiency. This paper has focused on sequential Monte Carlo approximations whose accuracy is difficult to analyze. However, it is important to note that there are high-dimensional, non-convex energy landscapes for which approximate sampling is provably efficient, even when optimization is provably NP-hard (94). But these results do not directly address the unreasonable in-practice effectiveness of sophisticated hybrids of data-driven and model-

driven Monte Carlo for high-dimensional probabilistic programs. There is a widespread need for new theory that can guide the design of real-time Monte Carlo approximations, accounting for design tradeoffs between latency, parallelism, and variance. One promising approach could be to try to extend recent spiking circuit formalisms from theoretical computer science to illuminate representational tradeoffs for parallel sampling circuits. Consider that given sufficient hardware, inference latency can be driven down to the number of serial steps needed to generate proposals, regardless of the number of variables being inferred or the complexity of the causal dependencies among them. But how do the mind and brain automatically constrain the structure of their generative models, so that "good enough, shallow enough" inference processes can be automatically generated?

Despite these open questions, the integrative theory introduced in this paper offers a candidate unifying framework for simultaneous reverse-engineering of the mind and brain at the computational, cognitive, and neural levels. It has survived an initial battery of empirical tests. We hope it enables neuroscientists and cognitive scientists to collaborate more closely with each other, and with artificial intelligence researchers, using a modeling formalism for intelligent computation that simultaneously addresses phenomenological, causal, and computational considerations. It also invites an intriguing question, in this time of excitement and concern about artificial intelligence: what useful intelligent systems can we build, if this brain-like model of computation is implemented using silicon that spikes millions of times faster than biological neurons?

Materials and Methods

Spiking Neural Monte Carlo Emulation in the Gen Probabilistic Programming System.. We have built an extension to the Gen probabilistic programming language (17) that allows scientists to run Monte Carlo inference algorithms in Gen probabilistic models and visualize the spiketrains that (one implementation of) Spiking Neural Monte Carlo may produce when running that inference algorithm on a given dataset. This extension also corrupts the probability calculations used to implement Monte Carlo inference algorithms, so that the probability computations are performed at low-precision, to perfectly match the scheme for obtaining probability-estimates used by our circuits for P-scoring, Q-scoring, and auto-normalization. To produce spiketrains corresponding to the Q-sampling, $1/Q$-scoring, and P-scoring for a given variable in a given particle on a given inference run, the emulator logs the noisy probability estimates it uses during inference. When asked to produce a spiketrain (for some subset of the neurons in cortex Layer 2-6 need for these operations), the emulator samples spiketrains consistent with these probability estimates and sampled values from the conditional distribution over spiketrains under the distribution over spiketrains implied by the circuits described in (the conditional distribution has a closed form due to the nice properties of Poisson Processes). The emulator can also produce spiketrains corresponding to the autonormalize-and-multiply operation. In effect this library will allow scientists to (1) experiment with how the quality of probabilistic inferences under a given model vary as the latency, assembly-sizes, and particle counts are varied to change the level of noise in the neural computations, and (2) produce spiketrains consistent with inference under given models, to compare to real biological activity to help them test the empirical predictions of the SNMC theory (and also to help them test what models and proposal distributions are actually present in the brains of different organisms).

Event-driven simulation of Spiking Neural Monte Carlo circuits.. To more thoroughly test the feasibility of Spiking Neural Monte Carlo, we have also implemented a compiler which can compile any model and algorithm in a restricted subset of the Gen probabilistic programming language into a spiking neural network which runs SNMC inference under the given model using the given inference algorithm. We used an event-driven simulator to simulate these neural networks and verify that they produce reasonable inferences in our simpler models. These simulations are run by feeding in spiking input events to the simulator at fixed intervals, to input the observed data to the neural circuit over time, and inference results are read out of the circuit in the Weighted Monte Carlo Spiking Code. The neural networks which are produced consist entirely of neurons whose spiking behavior is described by an inhomogenous Poisson Process; all the neurons we use have biologically realistic rates, except for some of the neurons used to implement logic-gating circuitry (which we expect is implemented using sub-neuronal mechanisms in the brain).

ACKNOWLEDGMENTS. The authors would like to thank Mehrdad Jazayeri and Melissa Kline for helpful comments on drafts.

1. Hv Helmholtz, *Treatise on Physiological Optics*. (Thoemmes Continuum, Bristol), (1856).
2. PS Laplace, *Philosophical Essay on Probabilities*. (1825).
3. S Russell, P Norvig, *Artificial Intelligence: A Modern Approach*. (Prentice Hall), (1995).
4. S Thrun, W Burgard, D Fox, *Probabilistic Robotics*. (MIT Press), (2005).
5. F Saad, Ph.D. thesis (Massachusetts Institute of Technology) (2022).
6. BM Lake, TD Ullman, JB Tenenbaum, SJ Gershman, Building machines that learn and think like people (2016).
7. F Lieder, TL Griffiths, Resource-rational analysis: Understanding human cognition as the optimal use of limited computational resources. *Behav. brain sciences* **43** (2020).
8. DC Knill, A Pouget, The bayesian brain: the role of uncertainty in neural coding and computation. *TRENDS Neurosci.* **27**, 712–719 (2004).
9. J Fiser, P Berkes, G Orbán, M Lengyel, Statistically optimal perception and learning: from behavior to neural representations. *Trends cognitive sciences* **14**, 119–130 (2010).
10. P Berkes, G Orbán, M Lengyel, J Fiser, Spontaneous cortical activity reveals hallmarks of an optimal internal model of the environment. *Science* **331**, 83–87 (2011).
11. SW Linderman, SJ Gershman, Using computational theory to constrain statistical models of neural data. *Curr. opinion neurobiology* **46**, 14–24 (2017).
12. WJ Ma, M Jazayeri, , et al., Neural coding of uncertainty and probability. *Annu. review neuroscience* **37**, 205–220 (2014).
13. N Goodman, V Mansinghka, DM Roy, K Bonawitz, JB Tenenbaum, Church: a language for generative models. *arXiv preprint arXiv:1206.3255* (2012).
14. B Milch, et al., 1 blog: Probabilistic models with unknown objects. *Stat. relational learning*, 373 (2007).
15. V Mansinghka, D Selsam, Y Perov, Venture: a higher-order probabilistic programming platform with programmable inference. *arXiv preprint arXiv:1404.0099* (2014).
16. VK Mansinghka, et al., Probabilistic programming with programmable inference in *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation*. pp. 603–616 (2018).
17. MF Cusumano-Towner, FA Saad, AK Lew, VK Mansinghka, Gen: a general-purpose probabilistic programming system with programmable inference in *Proceedings of the 40th acm sigplan conference on programming language design and implementation*. pp. 221–236 (2019).
18. N Gothoskar, et al., 3dp3: 3d scene perception via probabilistic programming in *NeurIPS*. (2021).
19. T Zhi-Xuan, J Mann, T Silver, J Tenenbaum, V Mansinghka, Online bayesian goal inference for boundedly rational planning agents. *Adv. Neural Inf. Process. Syst.* **33**, 19238–19250 (2020).
20. MF Cusumano-Towner, A Radul, D Wingate, VK Mansinghka, Probabilistic programs for inferring the goals of autonomous agents. *arXiv preprint arXiv:1704.04977* (2017).
21. BM Lake, R Salakhutdinov, JB Tenenbaum, Human-level concept learning through probabilistic program induction. *Science* **350**, 1332–1338 (2015).
22. TD Kulkarni, P Kohli, JB Tenenbaum, V Mansinghka, Picture: A probabilistic programming language for scene perception in *Proceedings of the ieee conference on computer vision and pattern recognition*. pp. 4390–4399 (2015).
23. N Roy, et al., From machine learning to robotics: Challenges and opportunities for embodied intelligence. *arXiv preprint arXiv:2110.15245* (2021).
24. M Cusumano-Towner, B Bichsel, T Gehr, M Vechev, VK Mansinghka, Incremental inference for probabilistic programs in *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation*. pp. 571–585 (2018).
25. AK Lew, M Cusumano-Towner, VK Mansinghka, Recursive monte carlo and variational inference with auxiliary variables. *arXiv preprint arXiv:2203.02836* (2022).
26. R Ranganath, S Gerrish, D Blei, Black box variational inference in *Artificial intelligence and statistics*. (PMLR), pp. 814–822 (2014).
27. D Wingate, ND Goodman, DM Roy, LP Kaelbling, JB Tenenbaum, Bayesian policy search with policy priors in *Twenty-second international joint conference on artificial intelligence*. (2011).
28. T Rainforth, TA Le, JW van de Meent, MA Osborne, F Wood, Bayesian optimization for probabilistic programs. *Adv. Neural Inf. Process. Syst.* **29** (2016).
29. M Riesenhuber, T Poggio, Hierarchical models of object recognition in cortex. *Nat. neuroscience* **2**, 1019–1025 (1999).
30. C Zhuang, et al., Unsupervised neural network models of the ventral visual stream. *Proc. Natl. Acad. Sci.* **118**, e2014196118 (2021).
31. DLK Yamins, et al., Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proc. Natl. Acad. Sci.* **111**, 8619–8624 (2014).

32. CF Cadieu, et al., Deep neural networks rival the representation of primate it cortex for core visual object recognition. *PLoS computational biology* **10**, e1003963 (2014).
33. R Rajalingham, A Piccato, M Jazayeri, The role of mental simulation in primate physical inference abilities. *bioRxiv* (2021).
34. GR Yang, MR Joglekar, HF Song, WT Newsome, XJ Wang, Task representations in neural networks trained to perform many cognitive tasks. *Nat. neuroscience* **22**, 297–306 (2019).
35. CL Alan L. Yuille, Limitations of deep learning for vision, and how we might fix them (2021).
36. U of California Los Angeles., Can artificial intelligence tell a teapot from a golf ball? (2019).
37. T Serre, Deep learning: The good, the bad, and the ugly. *Annu. review vision science* (2019).
38. JS Bowers, et al., Deep problems with neural network models of human vision (2022).
39. DL Yamins, JJ DiCarlo, Eight open questions in the computational modeling of higher sensory cortex. *Curr. Opin. Neurobiol.* **37**, 114–120 (2016) Neurobiology of cognitive behavior.
40. A Kurakin, I Goodfellow, S Bengio, Adversarial examples in the physical world (2016).
41. BBC, Uber in fatal crash had safety flaws say us investigators (2019).
42. Guardian, Tesla driver dies in first fatal crash while using autopilot mode (2016).
43. A Pouget, K Zhang, S Deneve, PE Latham, Statistically efficient estimation using population coding. *Neural Comput.* **10**, 373–401 (1998).
44. WJ Ma, JM Beck, PE Latham, A Pouget, Bayesian inference with probabilistic population codes. *Nat. neuroscience* **9**, 1432–1438 (2006).
45. D Pecevski, L Buesing, W Maass, Probabilistic inference in general graphical models through sampling in stochastic networks of spiking neurons. *PLoS computational biology* **7**, e1002294 (2011).
46. D Pecevski, W Maass, Learning probabilistic inference through spike-timing-dependent plasticity. *eneuro* **3** (2016).
47. L Buesing, J Bill, B Nessler, W Maass, Neural dynamics as sampling: a model for stochastic computation in recurrent networks of spiking neurons. *PLoS computational biology* **7**, e1002211 (2011).
48. R Legenstein, W Maass, Ensembles of spiking neurons with noise support optimal probabilistic inference in a dynamically changing environment. *PLOS Comput. Biol.* **10**, 1–27 (2014).
49. VK Mansinghka, EM Jonas, JB Tenenbaum, Stochastic digital circuits for probabilistic inference. *Massachussets Inst. Technol. Tech. Rep. MITCSAIL-TR* **2069** (2008).
50. EM Jonas, Ph.D. thesis (Massachusetts Institute of Technology) (2014).
51. V Mansinghka, E Jonas, Building fast bayesian computing machines out of intentionally stochastic, digital parts. *arXiv preprint arXiv:1402.4914* (2014).
52. P Redgrave, et al., Goal-directed and habitual control in the basal ganglia: implications for parkinson's disease. *Nat. Rev. Neurosci.* **11**, 760–772 (2010).
53. J Chrobak, G Buzsáki, Gamma oscillations in the entorhinal cortex of the freely behaving rat. *J. Neurosci.* **18**, 388–398 (1998).
54. SW Hughes, V Crunelli, Just a phase they're going through: The complex interaction of intrinsic high-threshold bursting and gap junctions in the generation of thalamic α and θ rhythms. *Int. J. Psychophysiol.* **64**, 3–17 (2007).
55. P Thaker, JB Tenenbaum, SJ Gershman, Online learning of symbolic concepts. *J. Math. Psychol.* **77**, 10–20 (2017).
56. J Tenenbaum, Rules and similarity in concept learning. *Adv. neural information processing systems* **12** (1999).
57. F Rieke, D Warland, R de Ruyter van Steveninck, W Bialek, *Spikes: Exploring the neural code*. (MIT Press), (year?).
58. MA Phillips, et al., A synaptic strategy for consolidation of convergent visuotopic maps. *Neuron* **71**, 710–724 (2011).
59. CF Stevens, Quantal release of neurotransmitter and long-term potentiation. *Cell* **72**, 55–63 (1993).
60. Y Zhang, et al., Asymmetric ephaptic inhibition between compartmentalized olfactory receptor neurons. *Nat. communications* **10**, 1–16 (2019).
61. A Blot, B Barbour, Ultra-rapid axon-axon ephaptic inhibition of cerebellar purkinje cells by the pinceau. *Nat. neuroscience* **17**, 289–295 (2014).
62. G Major, ME Larkum, J Schiller, Active properties of neocortical pyramidal neuron dendrites. *Annu. review neuroscience* **36**, 1–24 (2013).
63. G Major, A Polsky, W Denk, J Schiller, DW Tank, Spatiotemporally graded nmda spike/plateau potentials in basal dendrites of neocortical pyramidal neurons. *J. neurophysiology* **99**, 2584–2601 (2008).
64. AL Barth, JF Poulet, Experimental evidence for sparse firing in the neocortex. *Trends neurosciences* **35**, 345–355 (2012).
65. S Sakata, KD Harris, Laminar structure of spontaneous and sensory-evoked population activity in auditory cortex. *Neuron* **64**, 404–418 (2009).
66. NMM Amorim Da Costa, K Martin, Whose cortical column would that be? *Front. neuroanatomy*, 16 (2010).
67. A Naka, H Adesnik, Inhibitory circuits in cortical layer 5. *Front. neural circuits* **10**, 35 (2016).
68. XJ Wang, GR Yang, A disinhibitory circuit motif and flexible information routing in the brain. *Curr. opinion neurobiology* **49**, 75–83 (2018).
69. GR Yang, JD Murray, XJ Wang, A dendritic disinhibitory circuit mechanism for pathway-specific gating. *Nat. communications* **7**, 1–14 (2016).
70. A Bahramisharif, et al., Propagating neocortical gamma bursts are coordinated by traveling alpha waves. *J. Neurosci.* **33**, 18849–18854 (2013).
71. M Vinck, et al., Gamma-phase shifting in awake monkey visual cortex. *J. neuroscience* **30**, 1250–1257 (2010).
72. J Lisman, O Jensen, The theta-gamma neural code. *Neuron* **77**, 1002–1016 (2013).
73. C Andrieu, GO Roberts, The pseudo-marginal approach for efficient monte carlo computations. *The Annals Stat.* **37**, 697–725 (2009).
74. H Zhang, AJ Watrous, A Patel, J Jacobs, Theta and alpha oscillations are traveling waves in the human neocortex. *Neuron* **98**, 1269–1281 (2018).
75. J O'Keefe, ML Recce, Phase relationship between hippocampal place units and the eeg theta rhythm. *Hippocampus* **3**, 317–330 (1993).
76. MW Jones, MA Wilson, Phase precession of medial prefrontal cortical activity relative to the hippocampal theta rhythm. *Hippocampus* **15**, 867–873 (2005).
77. K Ok, K Liu, N Roy, Hierarchical object map estimation for efficient and robust navigation in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. (IEEE), pp. 1132–1139 (2021).
78. M Schrimpf, et al., Brain-score: Which artificial neural network for object recognition is most brain-like? *BioRxiv*, 407007 (2020).
79. I Yildirim, M Belledonne, W Freiwald, J Tenenbaum, Efficient inverse graphics in biological face processing. *Sci. advances* **6**, eaax5979 (2020).
80. AD Bolton, et al., Elements of a stochastic 3d prediction engine in larval zebrafish prey capture. *ELife* **8**, e51975 (2019).
81. H Zwaka, et al., Covert attention to obstacles biases escapes via the mauthner cell. *bioRxiv* (2022).
82. DF Goodman, R Brette, Brian: a simulator for spiking neural networks in python. *Front. neuroinformatics*, 5 (2008).
83. M Stimberg, R Brette, DF Goodman, Brian 2, an intuitive and efficient neural simulator. *eLife* **8**, e47314 (2019).
84. F Saad, M Cusumano-Towner, V Mansinghka, Estimators of entropy and information via inference in probabilistic models in *International Conference on Artificial Intelligence and Statistics*. (PMLR), pp. 5604–5621 (2022).
85. M Wick, A McCallum, Query-aware mcmc. *Adv. Neural Inf. Process. Syst.* **24** (2011).
86. FA Saad, MF Cusumano-Towner, U Schaechtle, MC Rinard, VK Mansinghka, Bayesian synthesis of probabilistic programs for automatic data modeling. *Proc. ACM on Program. Lang.* **3**, 1–32 (2019).
87. MA Fischler, RC Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* **24**, 381–395 (1981).
88. MF Cusumano-Towner, VK Mansinghka, Using probabilistic programs as proposals. *arXiv preprint arXiv:1801.03612* (2018).
89. A Lew, M Agrawal, D Sontag, V Mansinghka, Pclean: Bayesian data cleaning at scale with domain-specific probabilistic programming in *International Conference on Artificial Intelligence and Statistics*. (PMLR), pp. 1927–1935 (2021).
90. P Dayan, GE Hinton, Using expectation-maximization for reinforcement learning. *Neural Comput.* **9**, 271–278 (1997).
91. M Hoffman, A Doucet, N Freitas, A Jasra, Bayesian policy learning with trans-dimensional mcmc. *Adv. neural information processing systems* **20** (2007).
92. N Kantas, J Maciejowski, A Lecchini-Visintini, Sequential monte carlo for model predictive control in *Nonlinear model predictive control*. (Springer), pp. 263–273 (2009).
93. D Wingate, ND Goodman, DM Roy, LP Kaelbling, JB Tenenbaum, Bayesian policy search with policy priors in *Twenty-second international joint conference on artificial intelligence*. (2011).
94. YA Ma, Y Chen, C Jin, N Flammarion, MI Jordan, Sampling can be faster than optimization. *Proc. Natl. Acad. Sci.* **116**, 20881–20885 (2019).

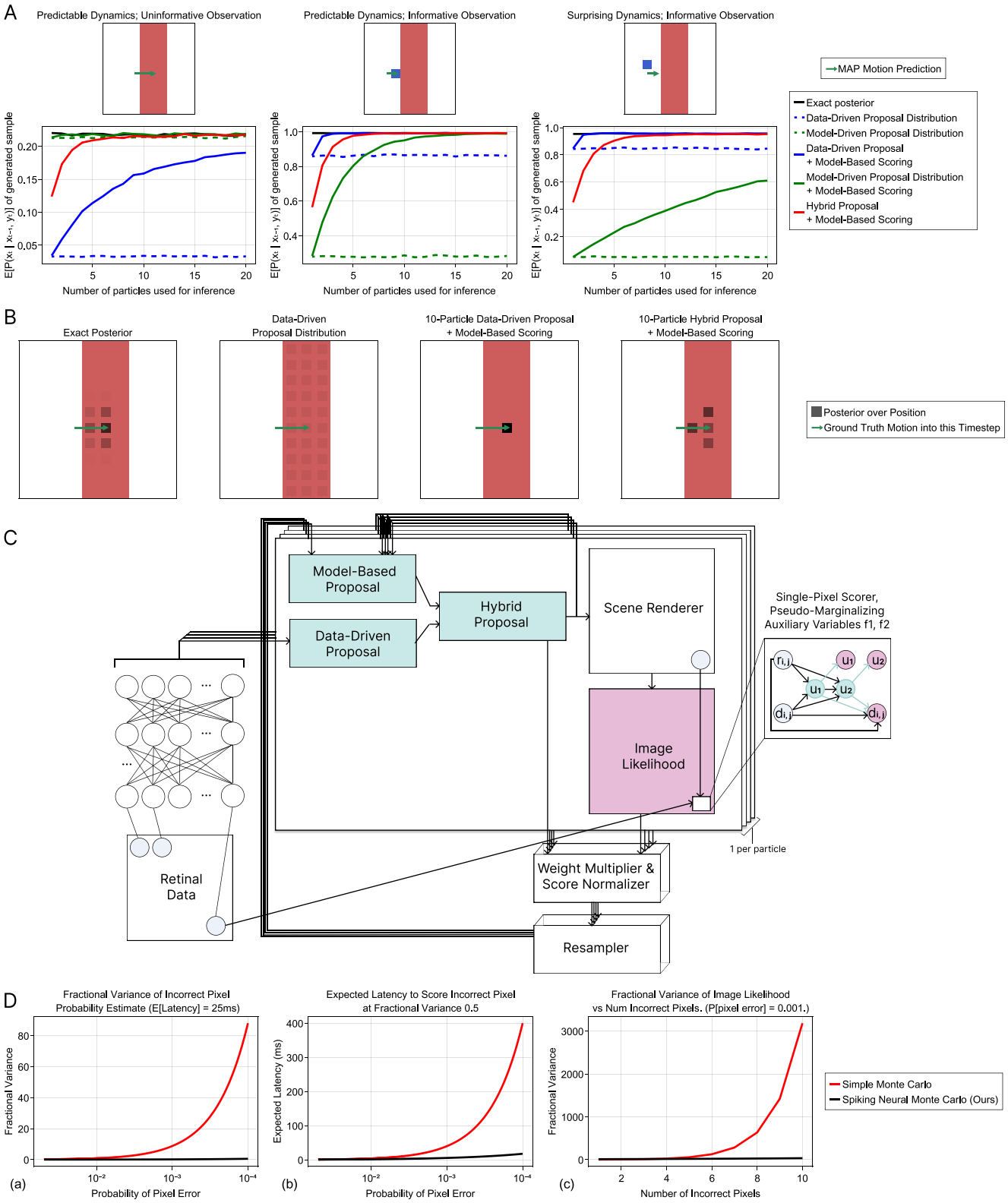


Fig. 7. Scaling up to mental physics simulation and inverse 2D graphics. (A) Hybrids of data-driven and model-driven inference scale better (top, red) than either data-driven or model-driven inference on its own. (B) The hybrid proposal closely approximates the exact Bayes filter, whereas the data-driven proposal struggles when the distinguished object is not visible. (C) The spiking neural Monte Carlo model combines a data-driven neural network (for bottom-up proposals) with top-down model-based inference. Accurate scoring of low-probability data (arising either when data is noisy or the internal model has large errors) is handled by nesting spiking neural Monte Carlo circuits for fast data-driven inference-based scoring (inset on right) within slower spiking neural Monte Carlo circuits for updating scene variables. (D) Inference-based scoring, in which spiking neural Monte Carlo inference over auxiliary variables is used to estimate rare event probabilities, scales to much lower probability data than simple Monte Carlo.

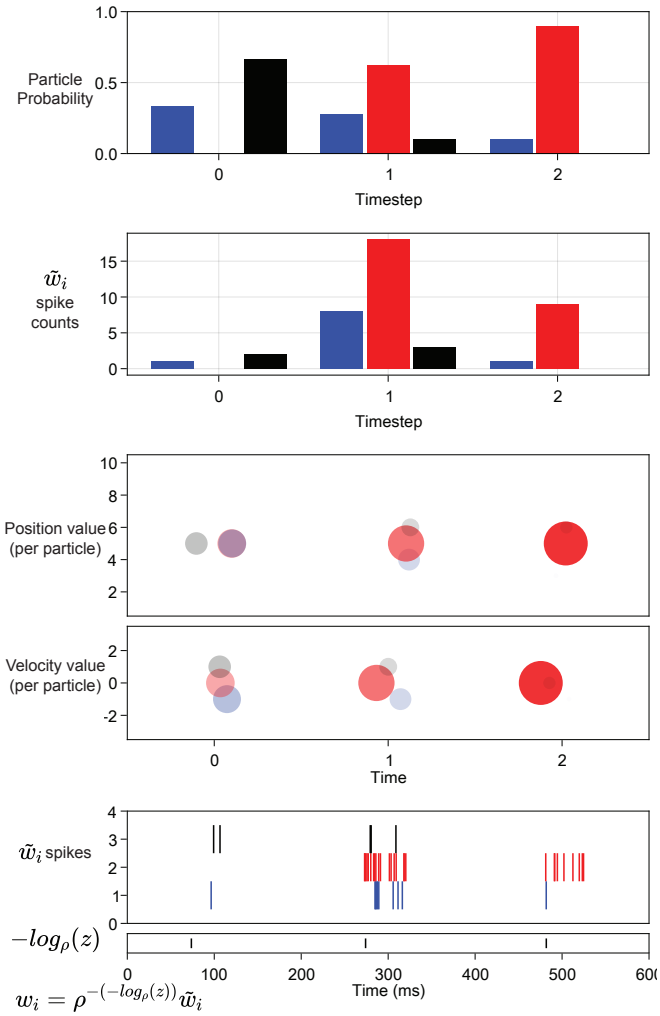


Fig. 8. Exact decoding of weights and probabilities (top) in a dynamically weighted Monte Carlo spiking code (bottom) requires a time-varying non-linear decoder. Spike rate does not always correlate with probability. For example, the probability of the red particle (top) increases significantly from timestep 1 to timestep 2, even when spike count (middle) drops so significantly that it is visible on the spike raster (bottom). Converting importance weights to probabilities requires normalization against the whole set of weights, i.e. $p^t(x_i) = f^t(w_i^t)$ with $f^t(w_i^t) = \frac{w_i^t}{\sum_j w_j^t}$. Thus if all other particles lose nearly all of their weight, the remaining particle's probability will increase, even if its weight drops somewhat significantly.

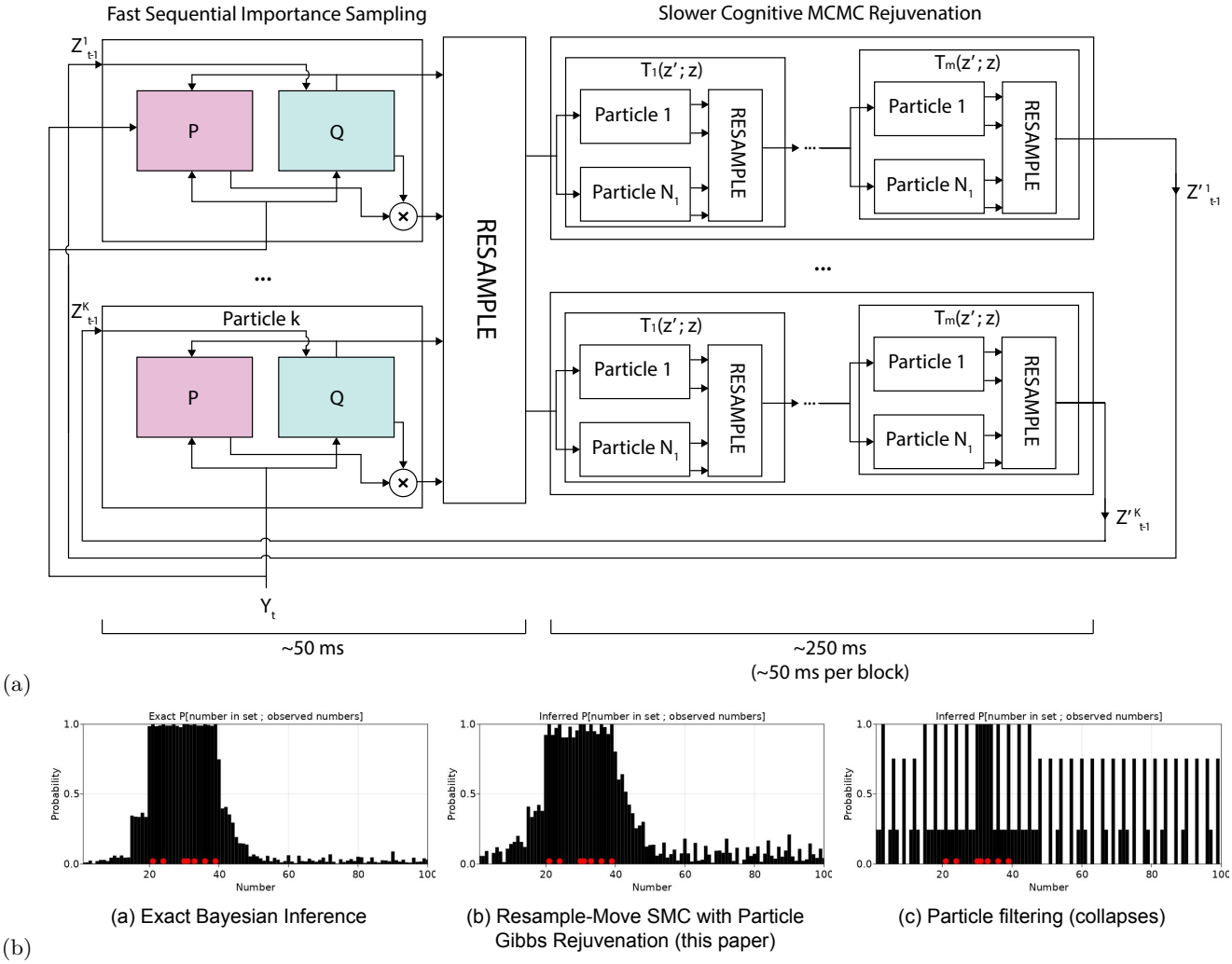


Fig. 9. Online Bayesian concept learning via hybrids of data-driven proposals with iterative, model-based MCMC. (top) Inference relies on fast data-driven proposals followed by model-based MCMC, using proposals that update small subsets of highly-coupled variables via sequential Monte Carlo with multiple particles. (bottom) This architecture makes it possible to explore a broad space of resource-rational, neurally mappable online approximations (55) to exact Bayesian inference (bottom, left) in a classic model of human concept learning (56). The spiking neural Monte Carlo circuit presented here (bottom, middle) better matches both exact Bayesian inference and the behavioral data from (55) than standard particle filtering (bottom, right), which fails to converge in this large hypothesis space.