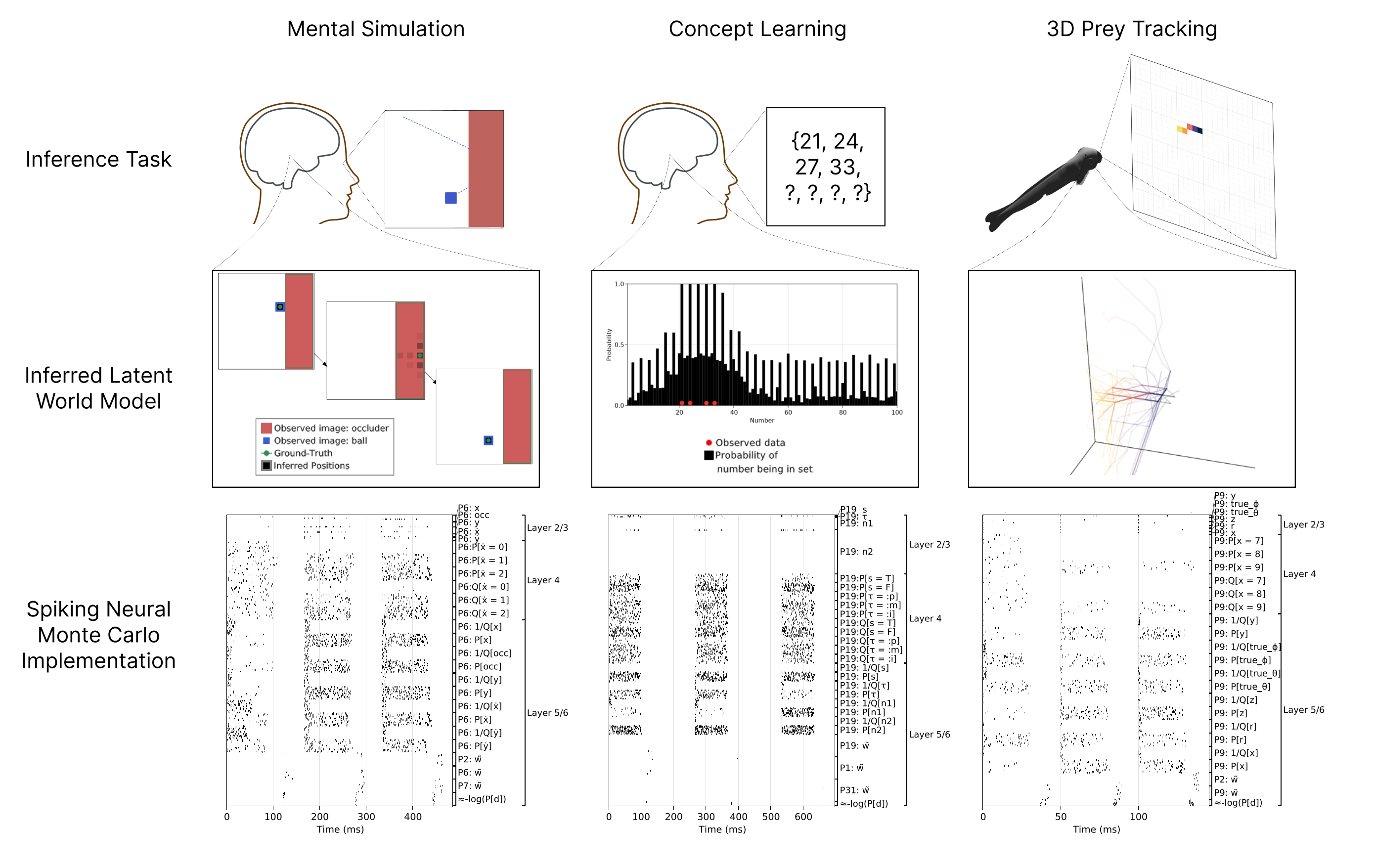
Brain computation as fast spiking neural Monte Carlo inference in probabilistic programs

George Matheos*, Andrew D Bolton*, McCoy Becker, Cameron Freer, Vikash Mansinghka

Introduction

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 can somehow approximate probabilistic inferences in complex probabilistic programs with many latent variables in real-time. Here we show how neurons could perform probabilistic inference, using massively parallel spiking assemblies to implement a novel neural coding scheme, called a dynamically weighted Monte Carlo spiking code. We prove that these assemblies generate approximate samples and make unbiased estimates of probabilities and importance weights, enabling sound approximate inference. Sampled latent variables are sparsely coded, but probabilities and weights are densely coded, and can be read via time-varying, divisively normalized decoding of the dense spiking from specific sub-populations. We show how to implement data-driven artificial neural networks for making fast, bottom-up proposals that are scored and corrected using a structured generative model, yielding new hybrids of distributed and localized neural representations. These spiking neural Monte Carlo architectures scale exponentially better than probabilistic population codes, and are neurally mappable, but unlike deep learning models, they also enable sound implementations of state-of-the-art model-driven AI architectures and inference processes from Bayesian cognitive science. We demonstrate generality by providing spiking circuits for probabilistic program models of visual prey tracking by larval zebrafish, mental physics simulation by primates, and human concept learning. We also present empirical support for this theory, confirming predictions for neural connectivity, coding, and dynamics using data from multiple brain regions and model organisms.

SNMC scales to real time perception and cognition



	Latent Variables	Observed Variables	Size of Spiking Neural Representation Weighted Monte Carlo (this paper)	 ENS Codes, Standard PPCs
1D object tracking	$\{x_t, \dot{x}_t\}_t$	$\{d^x_t\}_t$	Sparse: 27 Dense: 5	Dense: 140
2D object tracking	$\{x_t, y_t, \dot{x}_t, \dot{y}_t\}_t$	$\{d^x_t, d^y_t\}_t$	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	$\{x_t, y_t, z_t, \dot{x}_t, \dot{y}_t, \dot{z}_t, r_t, \phi_t, \theta_t\}_t$	$\{d^\phi_t, d^\theta_t\}_t$	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}

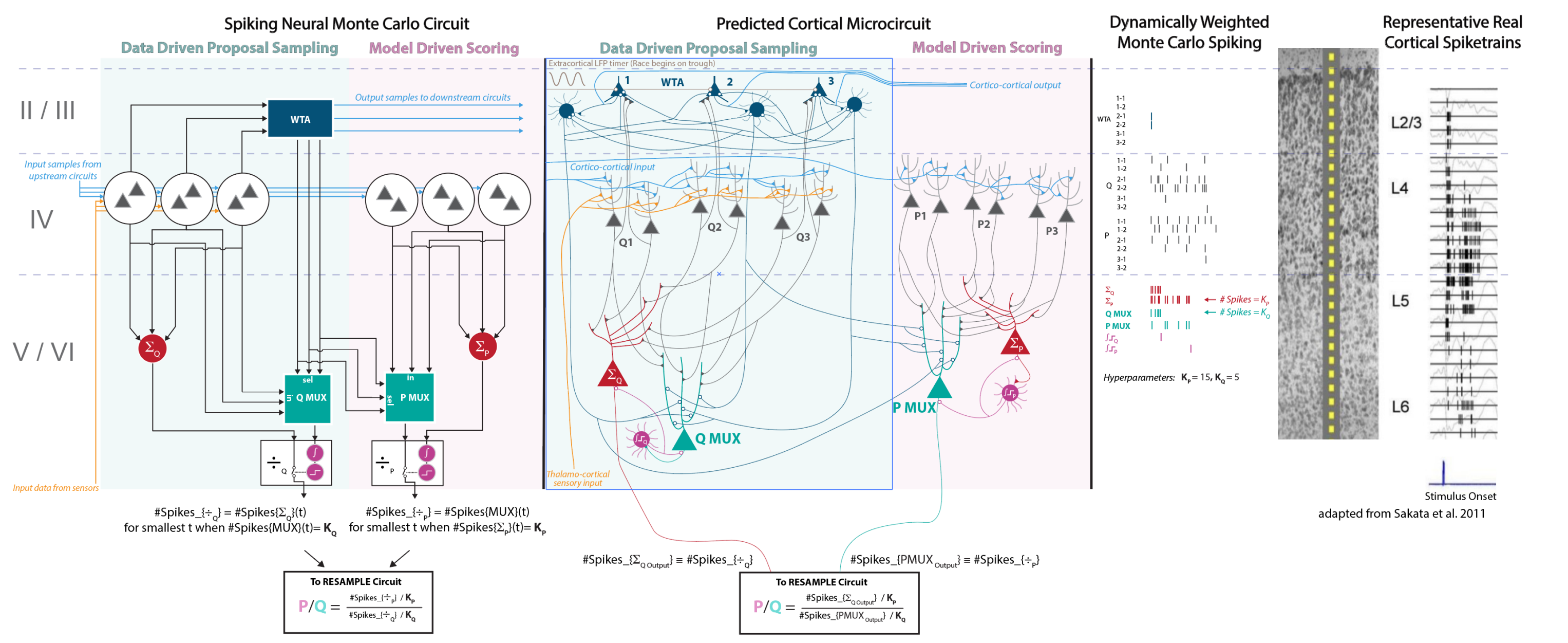
Spiking Neural Monte Carlo requires exponentially fewer neurons than standard probabilistic population codes and ENS spiking codes. For low-dimensional probabilistic programs that only make a small number of latent choices, the difference can be modest in absolute terms. As the number of latent variables 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. (Ma, Beck, et al 2006, Legenstein and Maass 2014).

Spiking Neural Monte Carlo circuits

Micro-scale Predictions of SNMC

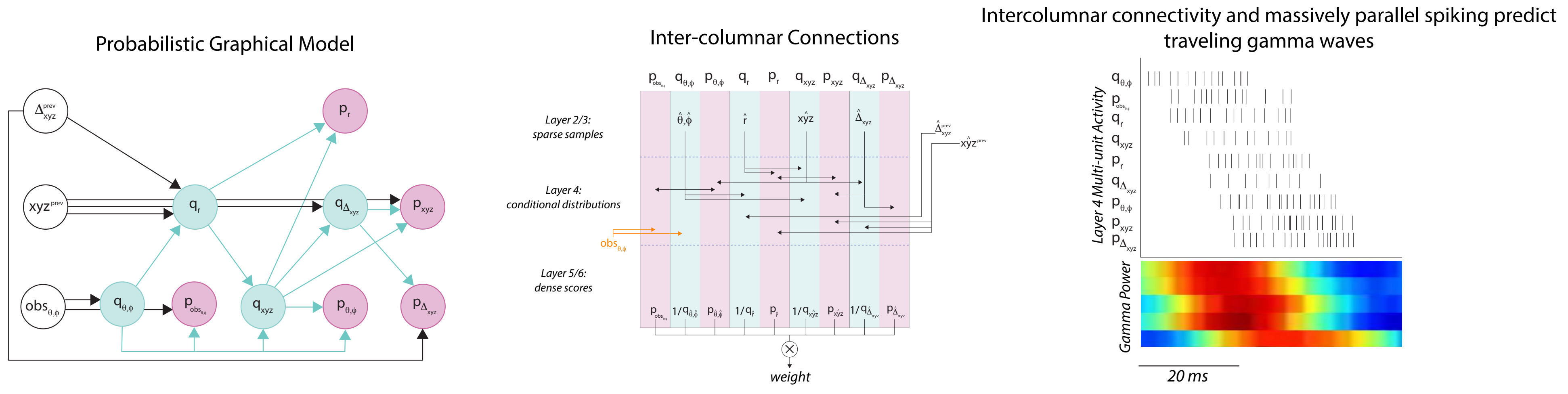
SNMC importance sampler circuit topology, sorted given known sensory (thalamo-cortical) L4 input according to known intra-layer cortical connectivity, predicts multiple findings in hodology, synaptic physiology, and extracellular spike & field electrophysiology:

1. Observed L2/3 sparse and L4/5 dense spiking and relative timing
2. The existence of thalamo-cortical L4 inputs
3. The existence of specific cell types & synaptic physiology in L4/5



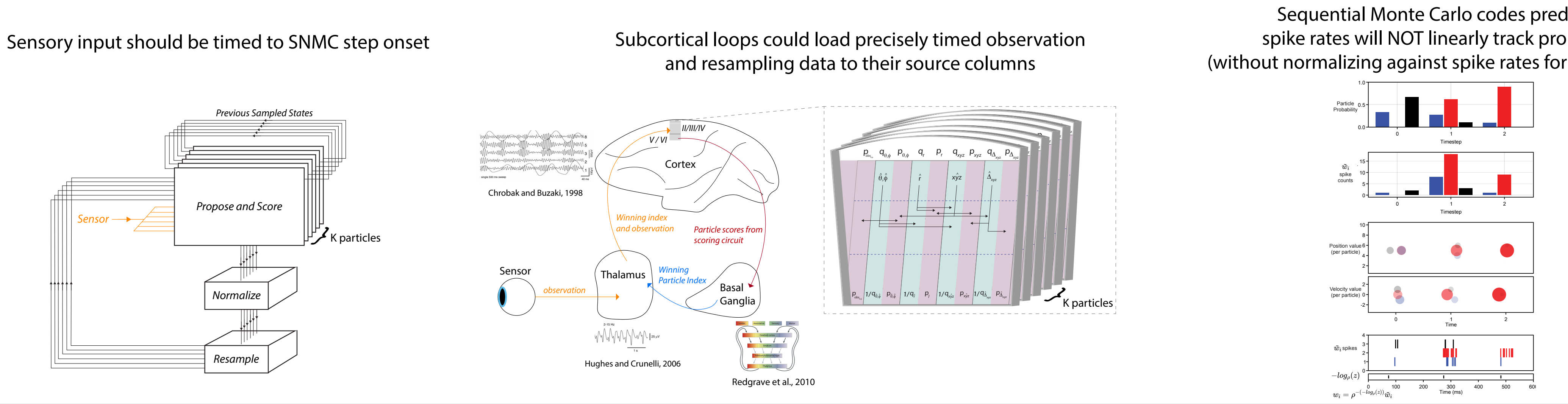
Meso-scale Predictions of SNMC

Successive sample-score epochs, whose order is defined by dependency structure, produce phase shifted bursts of activity in neighboring columns

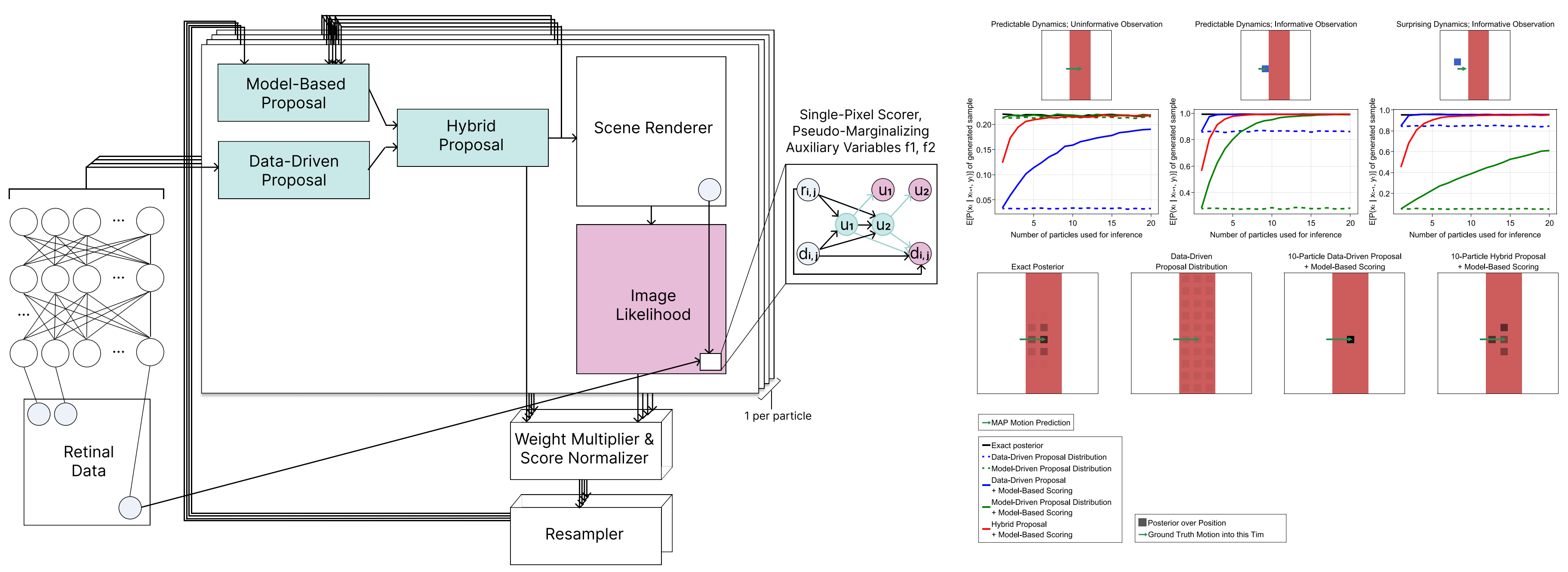


Macro-scale Predictions of SNMC

Topology and timing of sequential Monte Carlo - loading data, proposing variables, scoring variables, and resampling particles - predicts parallel cortico-subcortical loops for each particle, synchronized within particles (across columns) and across columns, via cortico-thalamic theta rhythms



SNMC model of primate physical scene understanding, including data-driven ANN proposals

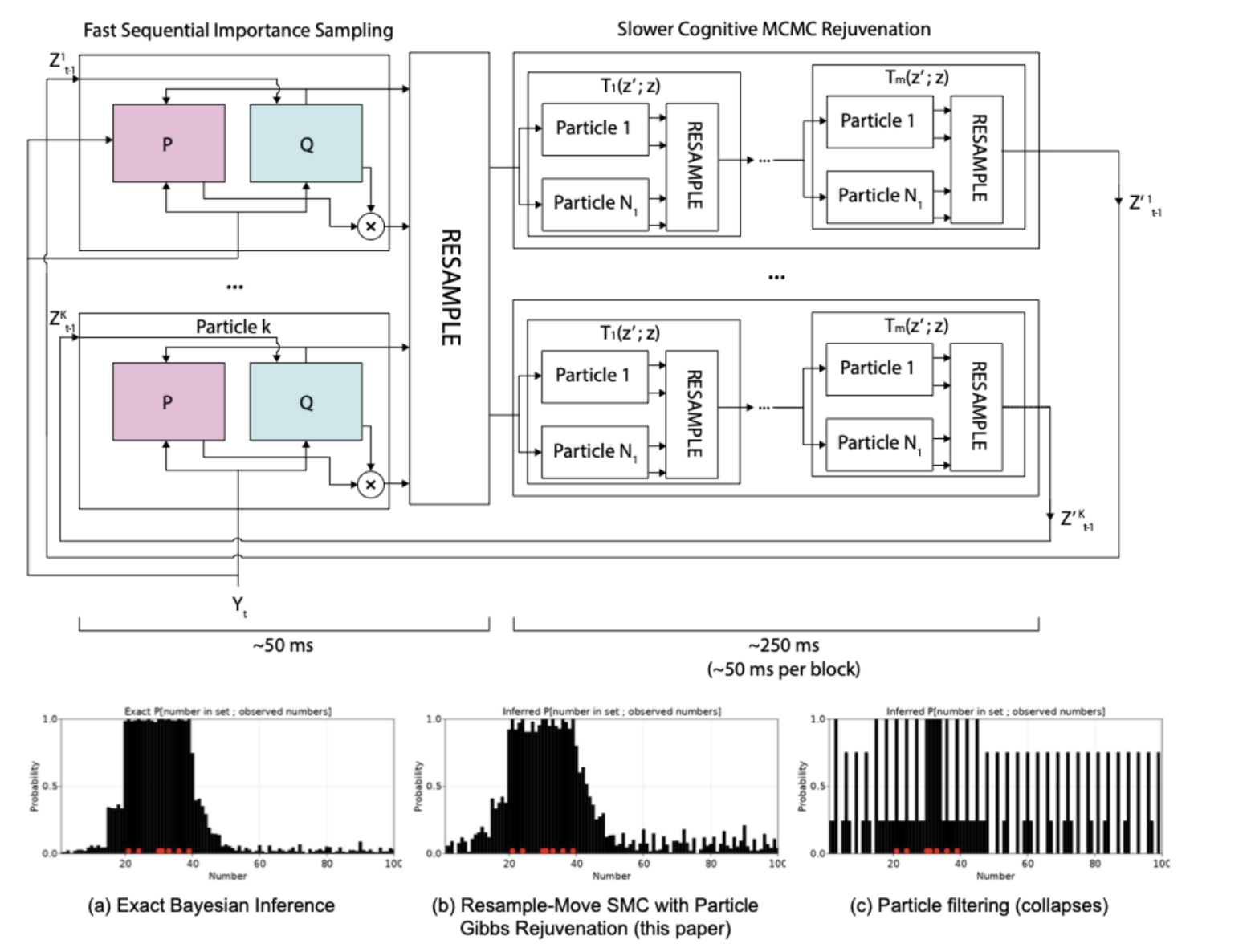


Benefits over ANNs alone: improved robustness, due to top-down model-based scoring, plus a model for how probabilistic programs in the brain could generate training data for data-driven ANNs

Supporting the Embodied Intelligence Mission and the Development of Intelligence Mission

Model Based on Rishi Rajalingham, Aida Piccato, Mehrdad Jazayeri 2021

SNMC model of human concept learning via data-driven cognitive MCMC



(a) Exact Bayesian Inference

(b) Resample-Move SMC with Particle Gibbs Rejuvenation (this paper)

(c) Particle filtering (collapses)

Benefits over vanilla SMC / particle filtering: robust convergence even for high dimensional problems & unlikely data, due to data-driven proposals and MCMC updates (Gilks and Beruzini 2001)

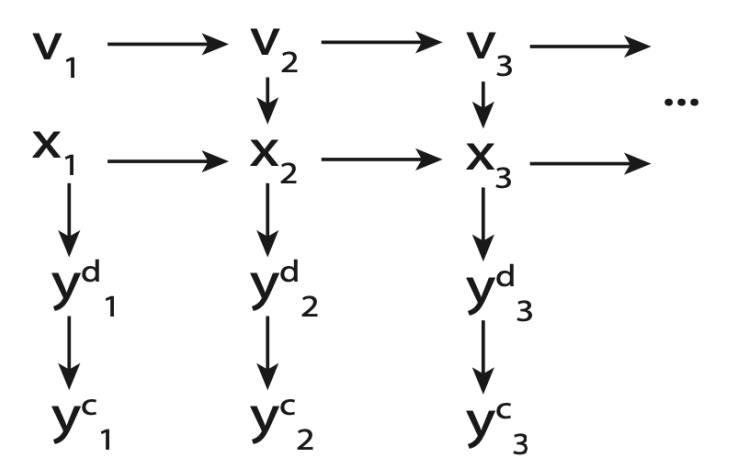
Supporting the Supporting the Development of Intelligence Mission

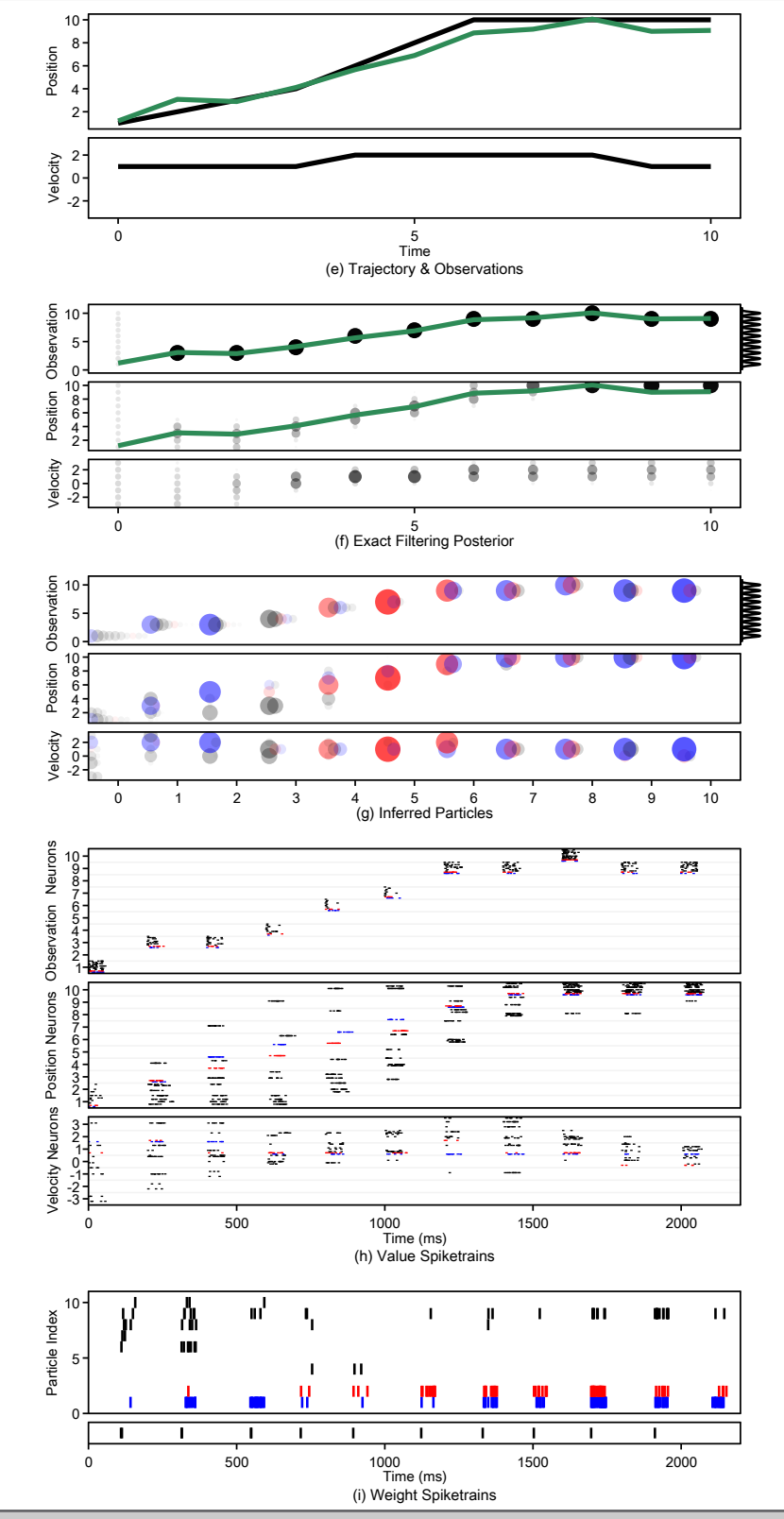
Model Based On Pratiksha Thaker, Joshua B. Tenenbaum, Samuel J. Gershman 2017; Tenenbaum 1999

Constructing SNMC circuits from Gen probabilistic programs

```
@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





Math

1 Equations for Spike Counts

1.1 P scoring

Say random variable X takes values in $\{1, 2, \ldots, |X|\}$. To represent the distribution $P(X|U)$, there are $|X|$ assemblies. Given the value u of X's parent variable U, the ith assembly spikes as a Poisson process with rate

$$\lambda^i_P = \lambda \times P(X = i|u)$$

After τ seconds, the number of spikes from the ith assembly is

$$N^i_P(\tau) \overset{\text{i.i.d.}}{\sim} \text{Poisson}(\lambda \times P(X = i|u) \times \tau)$$

For sampled value $x \in \{1, 2, \ldots, |X|\}$, we estimate $P(X = x|u)$ using

$$p = \frac{N^x_P(\tau)}{\sum_{i=1}^{|X|} N^i_P(\tau)} \approx P(X = x|u)$$

For any integer hyperparameter K_P, (e.g. $K_P = 15$), If we take $\tau = \tau_P$ where

$$\tau_P = \text{ the smallest time } \tau \text{ s.t. } \sum_{i=1}^{|X|} N^i_P(\tau) = K_P$$

Then

$$E[p] = P(X = x|u)$$

1.2 Q sampling and scoring

To represent the distribution $Q(X|V)$, there are $|X|$ assemblies. Given the value v of V, the ith assembly spikes as a Poisson process with rate

$$\lambda^i_Q = \lambda \times Q(X = i|v)$$

For $j = 1, 2, \ldots$, the index of the assembly which emitted the jth spike, A_j, is uniformly sampled from Q:

$$A_j \overset{\text{i.i.d.}}{\sim} Q(X|v)$$

In particular, $A_1 \sim Q(X|v)$.

After τ seconds, the number of spikes from the ith assembly is

$$N^i_Q(\tau) \overset{\text{i.i.d.}}{\sim} \text{Poisson}(\lambda \times Q(X = i|v) \times \tau)$$

For sampled value $x \in \{1, 2, \ldots, |X|\}$, we estimate $\frac{1}{Q(X=x|v)}$ using

$$q^{-1} = \frac{\sum_{i=1}^{|X|} N^i_Q(\tau)}{N^x_Q(\tau)} \approx Q(X = x|v)$$

For any integer hyperparameter K_Q, (e.g. $K_Q = 5$), If we take $\tau = \tau_Q$ where

$$\tau_Q = \text{ the smallest time } \tau \text{ s.t. } N^x_Q(\tau) = K_Q$$

Then

$$E[q^{-1}] = \frac{1}{Q(X = x|u)}$$

2 Pseudo-Marginal Importance Sampling for One Variable

A joint probability distribution

$$P(X, Y)$$

is encoded in the spiking neural network's parameters, as is a proposal kernel

$$Q(X; Y)$$

2.1 Importance Sampling

Given observed value $Y = y$, the importance sampling uses N copies of a sampling circuit to sample, for each $i \in \{1, 2, \ldots, N\}$,

$$x^i \overset{\text{i.i.d.}}{\sim} Q(\cdot; y)$$

Each sampling circuit is paired with a scoring circuit which stochastically computes values w_i such that

$$E[w^i] = \frac{P(x^i, y)}{Q(x^i; y)}$$

Let $(\tilde{w}^i)_{i=1}^N$ be the normalized weights

$$\tilde{w}^i = \frac{w^i}{\sum_{j=1}^N w^j}$$

2.2 Convergence Guarantees

Under standard technical conditions on Q, the weighted particle cloud $\{(x^i, \tilde{w}^i)\}_{i=1}^N$ converges to the posterior distribuion under P:

$$\sum_{i=1}^N \tilde{w}^i \delta_{x^i} \underset{N\to\infty}{\overset{w}{\to}} P(X|y)$$

For any finite N, the unnormalized particle weights form an unbiased estimate of $P(y)$:

$$E[\frac{1}{N}\sum_{i=1}^N w^i] = P(y)$$

These guarantees are standard in importance sampling. SNMC differs from standard IS by using stochastic, low-precision estimates w^i of $\frac{P(x^i,y)}{Q(x^i;y)}$. By applying Monte-Carlo analysis techniques from [5], we prove that these formal guarantees still hold when using low-precision neural estimates of importance weights.

2.3 Unbiased Weighting in SNMC

In Spiking Neural Monte Carlo, we take

$$w^i = p^i(q^i)^{-1}$$

where p^i and $(q^i)^{-1}$ are stochastically computed by a neural circuit in such a way that

$$E[p^i] = P(x^i, y), \; E[q^i] = \frac{1}{Q(x^i, y)}$$

3 Using Artificial Neural Networks within Spiking Neural Monte Carlo

For an animal to efficiently track an object moving in 2D, using images on a 3x3 retina, we need an efficient proposal distribution

$$Q(X_t, Y_t; X_{t-1}, Y_{1-t}, V^x_{t-1}, V^y_{t-1}, \text{Grid}_t)$$

where

$$\text{Grid}_t = \begin{bmatrix} \text{Grid}_t[1,1] & \text{Grid}_t[1,2] & \text{Grid}_t[1,3] \\ \text{Grid}_t[2,1] & \text{Grid}_t[2,2] & \text{Grid}_t[2,3] \\ \text{Grid}_t[3,1] & \text{Grid}_t[3,2] & \text{Grid}_t[3,3] \end{bmatrix}$$

To encode such a proposal distribution efficiently, can use an artificial neural network

$$f_\theta : \mathcal{O}_X \times \mathcal{O}_Y \times \mathcal{O}_{V^x} \times \mathcal{O}_{V^y} \times \mathcal{O}_X \times \mathcal{O}_{\text{Grid}} \to \mathbb{R}^{|X|} \times \mathbb{R}^{|Y|}$$

where $\mathcal{O}_V$ denotes the space of one-hot encodings of variable V. f_θ accepts a one-hot encoding of $X_{t-1}, Y_{1-t}, V^x_{t-1}, V^y_{t-1}$, and Grid_t, and outputs distributions over X_t and Y_t:

$$Q(X_t = x; x_{t-1}, x_{1-t}, v^x_{t-1}, v^y_{t-1}, \text{grid}_t) \propto f_\theta(x_{t-1}, x_{1-t}, v^x_{t-1}, v^y_{t-1}, \text{grid}_t)_x$$

$$Q(Y_t = y; x_{t-1}, x_{1-t}, v^x_{t-1}, v^y_{t-1}, \text{grid}_t) \propto f_\theta(x_{t-1}, x_{1-t}, v^x_{t-1}, v^y_{t-1}, \text{grid}_t)_{|X|+y}$$

This is implemented by copying the spiking rates from the last layer of the neural network to the assemblies in the proposal distributions in Spiking Neural Monte Carlo circuits for sampling and scoring X_t and Y_t.

f_θ is implemented in the spiking network as a fully-connected network of neurons with parameters θ encoded in neuron biases and connection strengths, using neuron-rates as approximations of real values. Our work does not analyze the impact of this approximation on ANN performance precision, nor how the brain could implement backpropagation.

4 Resample-Move SMC with Particle Gibbs Rejuvenation

4.1 Resample-Move Sequential Monte Carlo

Dynamic Probabilistic Programs. Resample-Move Sequential Monte Carlo runs inference in a dynamic probabilistic program

$$P(x_{0:T}, y_{0:T}) = P_0(x_0)P_y(y_0|x_0)\prod_{t=1}^T P_\to(x_t|x_{t-1})P_y(y_t|x_t)$$

Each x_t and y_t may be a collection of values for many different variables (e.g. y_t might be a grid giving the intensity of color at every point on an animal's retina).

Resample-Move SMC. Resample-Move SMC interweaves Importance Sampling steps with Resampling steps and Particle Gibbs MCMC "rejuvenation" steps. Importance sampling uses two data-driven proposal distributions $Q_0(X_0|y_0)$ and $Q_\to(X_t|x_{t-1}y_t)$

Initial importance sampling. At $t = 0$, N initial particles are generated and weighted. For $i \in \{1, 2, \ldots, N\}$:

$$X^i_0 \sim Q_0(\cdot|y_0)$$

$$w^i_0 \approx \frac{P_0(x_0)P_y(y_0|x_0)}{Q_0(x_0; y_0)}$$

Resampling. At each timestep $t > 0$, particles from the previous step are resampled, by selecting N "ancestor" indices

$$A^i_t \sim \text{Categorical}(\tilde{w}^1_{t-1}, \tilde{w}^2_{t-1}, \ldots, \tilde{w}^N_{t-1})$$

Here, $\tilde{w}^i_t$ is the normalized weight

$$\tilde{w}^i_t := \frac{w^i_t}{\sum_{j=1}^N w^j_t}$$

MCMC Rejuvenation. Each resampled particle is optionally rejuvenated by running a MCMC M on it:

$$x'^{\,i}_{t-1} \sim M(\cdot; x^{A_i}_{t-1}, y_t)$$

Particle extension. Finally, each particle is extended to the next timestep, and weighted:

$$x^i_t \sim Q_\to(\cdot; x'^{\,i}_{t-1}, y_t)$$

$$w^i_t \approx \frac{P_\to(x^i_t|x'^{\,i}_{t-1})P_y(y_t|x^i_t)}{Q_\to(x^i_t; x'^{\,i}_{t-1}, y_t)}$$

4.2 Particle Gibbs Rejuvenation

One family of MCMC kernels is *Particle Gibbs*. Spiking Neural Monte Carlo can be used to implement any Particle Gibbs algorithm; here we present the simplest class of Particle Gibbs algorithms.

Proposing modifications to the latent state. Given current latent state x and evidence y, Particle Gibbs samples $N_M - 1$ proposed modifications to the value using a proposal distribution Q_M. We take $x^1 := x$, and for $i \in \{2, \ldots, N_M\}$:

$$x^i \overset{\text{i.i.d.}}{\sim} Q_M(\cdot; x, y)$$

There are restrictions on the family of valid proposal distributions Q_M. One key restriction is that Q_M may not condition on the value of a variable v in state x if Q_M can propose to change the value of v.

Selecting the next state.

Importance weights are computed:

$$w^i = \frac{P(x^i, y)}{Q_M(x^i; x)}$$

And a particle index is sampled:

$$i^* \sim \text{Categorical}(\tilde{w}^1, \tilde{w}^2, \ldots, \tilde{w}^{N_M})$$

where

$$\tilde{w}^i = \frac{w^i}{\sum_{j=1}^{N_M}}$$

Particle-Gibbs returns x^{i^*} as the next MCMC state:

$$x' := x^{i^*}$$

Stationarity. Particle-Gibbs is stationary for $P(\cdot|y)$, meaning that

$$x \sim P(\cdot|y), x' \sim M(\cdot; x, y) \implies x' \sim P(\cdot|y)$$

When $N_M = 2$, this class of Particle Gibbs reduces to a variant of Metropolis-Hastings.