Efficient Probabilistic Inference in the Quest for Physics Beyond the Standard Model

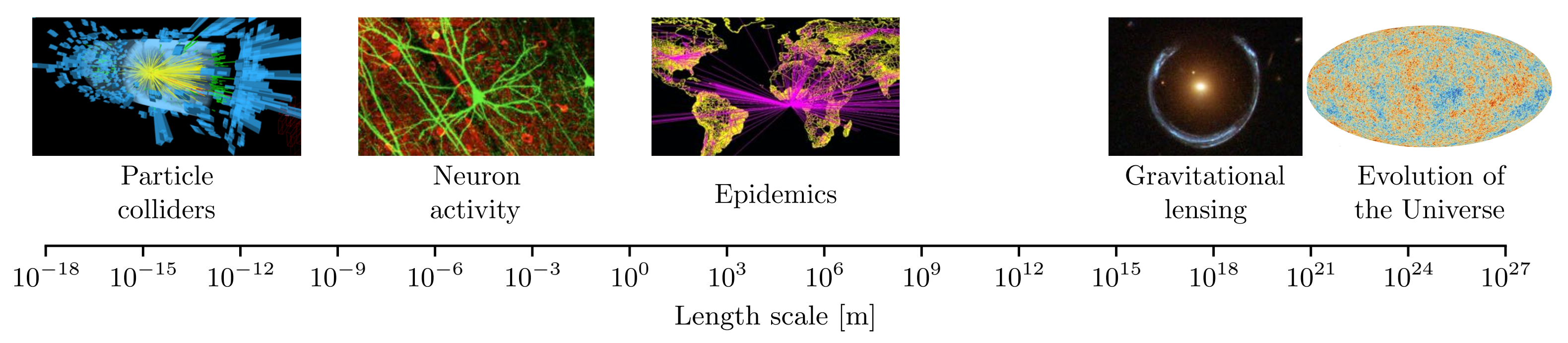
Atılım Güneş Baydin,[1] Lukas Heinrich,[2] Wahid Bhimji,[3] Lei Shao,[4] Saeid Naderiparizi,[5] Andreas Munk,[5] Jialin Liu,[3]
Bradley Gram-Hansen,[1] Gilles Louppe,[6] Lawrence Meadows,[4] Philip Torr,[1] Victor Lee,[4] Prabhat,[3] Kyle Cranmer,[7] Frank Wood[5]
[1]University of Oxford, [2]CERN, [3]Lawrence Berkeley National Lab, [4]Intel Corporation, [5]University of British Columbia, [6]University of Liege, [7]New York University

Summary

- Stochastic **simulators are equivalent to probabilistic programs** if we could capture all randomness and perform conditioning. We introduce a framework that enables this for any existing large-scale simulator code base in any language.
- We demonstrate it in a state-of-the-art particle physics simulator at a scale of 1M lines of C++ code and more than 25k latent variables.
- Amortized inference using LSTM-based dynamic proposal networks improves sample efficiency and makes simulator-scale inference feasible.

Probabilistic Programming in Existing Simulators

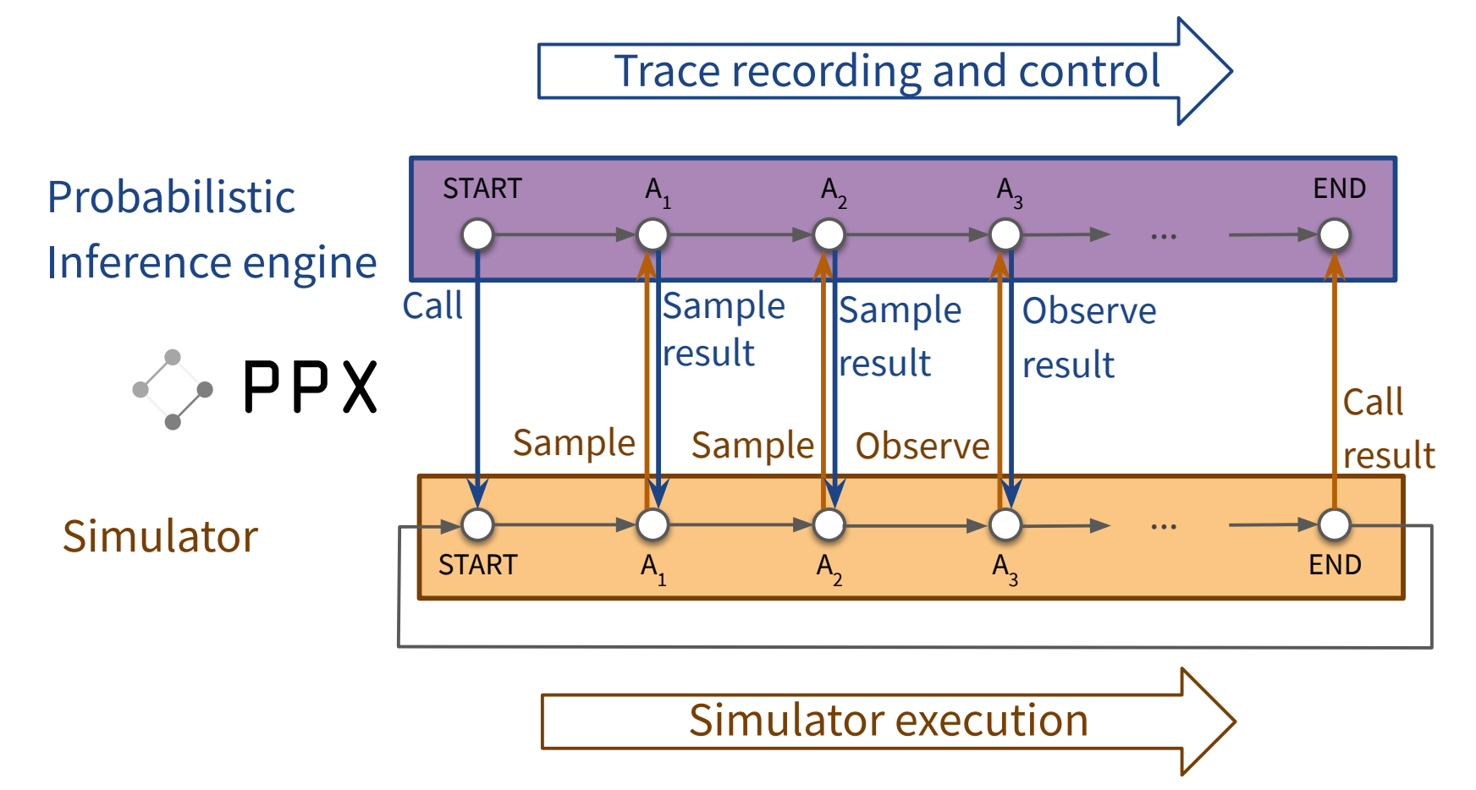
Probabilistic programming allows one to define probabilistic models using general-purpose programming languages [1]. A program defines the joint prior $p(\mathbf{x},\mathbf{y}) := \prod_{t=1}^{T} f_{a_t}(x_t|x_{1:t-1}) \prod_{n=1}^{N} g_n(y_n|x_{\prec n})$ of latents $\mathbf{x}$ and observables $\mathbf{y}$, and inference approximates the posterior $p(\mathbf{x}|\mathbf{y})$ given observations $\mathbf{y}$.

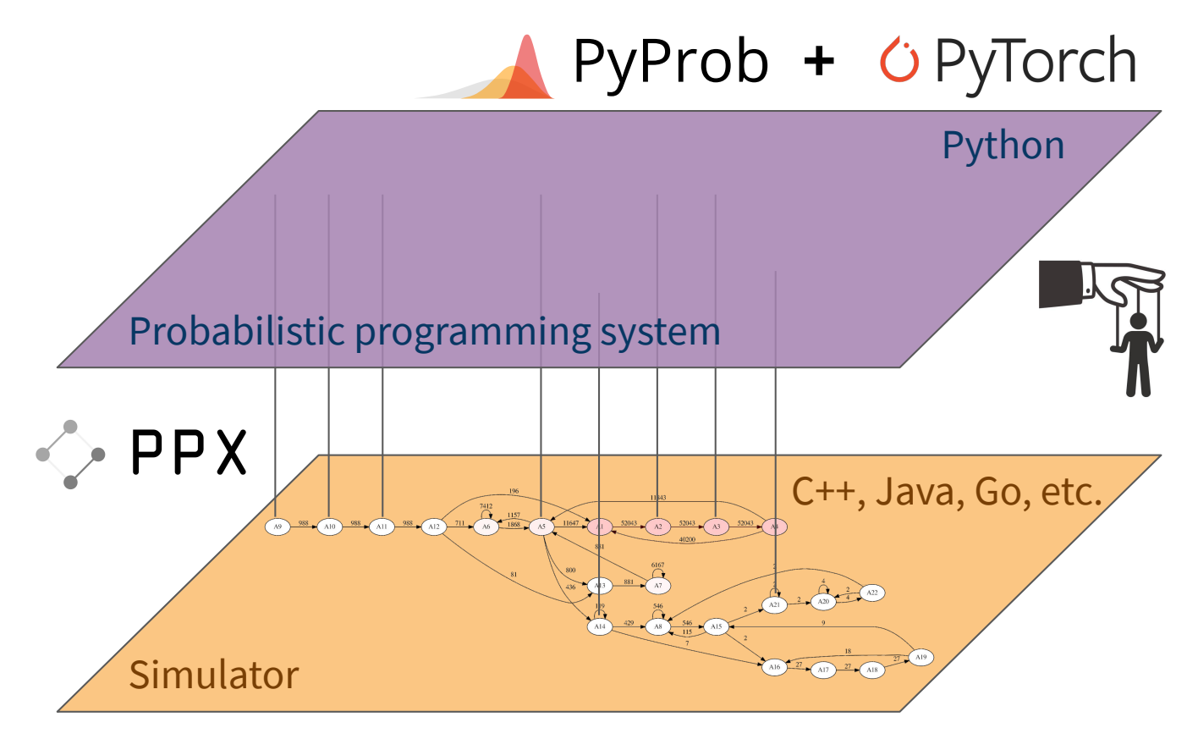


Many domains of science have complex simulators that describe the current best understanding of phenomena [2]. Building on our previous preliminary work [3], our work **enables the execution of existing, large-scale simulator codes as probabilistic programs with minimal alteration.** We do this by catching all random number calls as an *execution trace* and conditioning on observed data.

How Do We Record Execution Traces?

We introduce PyProb, a new universal probabilistic programming system designed to work with existing simulators in any language. It supports distributed training and inference in simulators using PyTorch MPI.





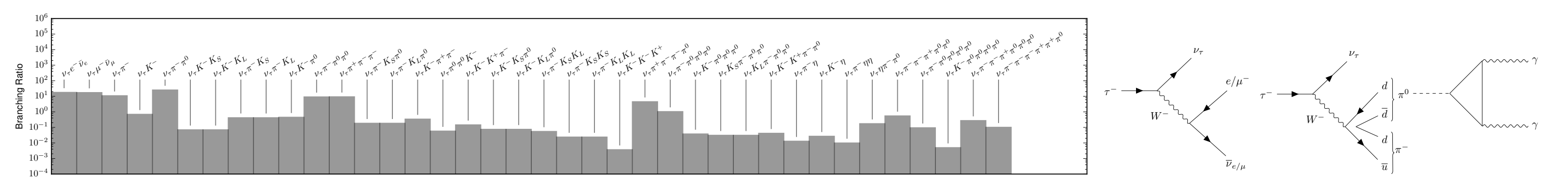
PyProb connects to simulators using the new Probabilistic Programming eXecution (PPX) protocol based on Flatbuffers (supporting C++, C#, Go, Java, JavaScript, PHP, Python, TypeScript, Rust, Lua, and others). We release PPX as a separate project and a probabilistic programming analog of ONNX.

PPX **replaces simulator's random number generator** so that we can

- record an execution trace $\{(x_t)_{t=1}^T, (y_n)_{n=1}^N\}$ sampled from simulator prior $p(\mathbf{x},\mathbf{y})$
- or guide simulator execution by sampling from proposals $q(x_t|\cdot)$ at runtime.

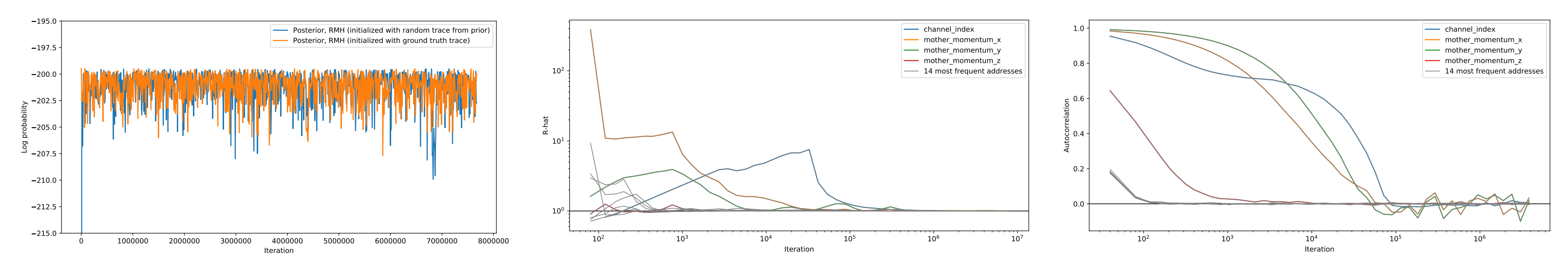
Use Case: Particle Physics

We study τ (tau) lepton decay in the Standard Model of Particle Physics, which is a key ingredient in establishing the properties of the Higgs boson. We use the state-of-the-art Sherpa simulator coupled to a fast calorimeter implementation, both in C++, and connect these to PyProb.



Markov Chain Monte Carlo (MCMC) Baseline

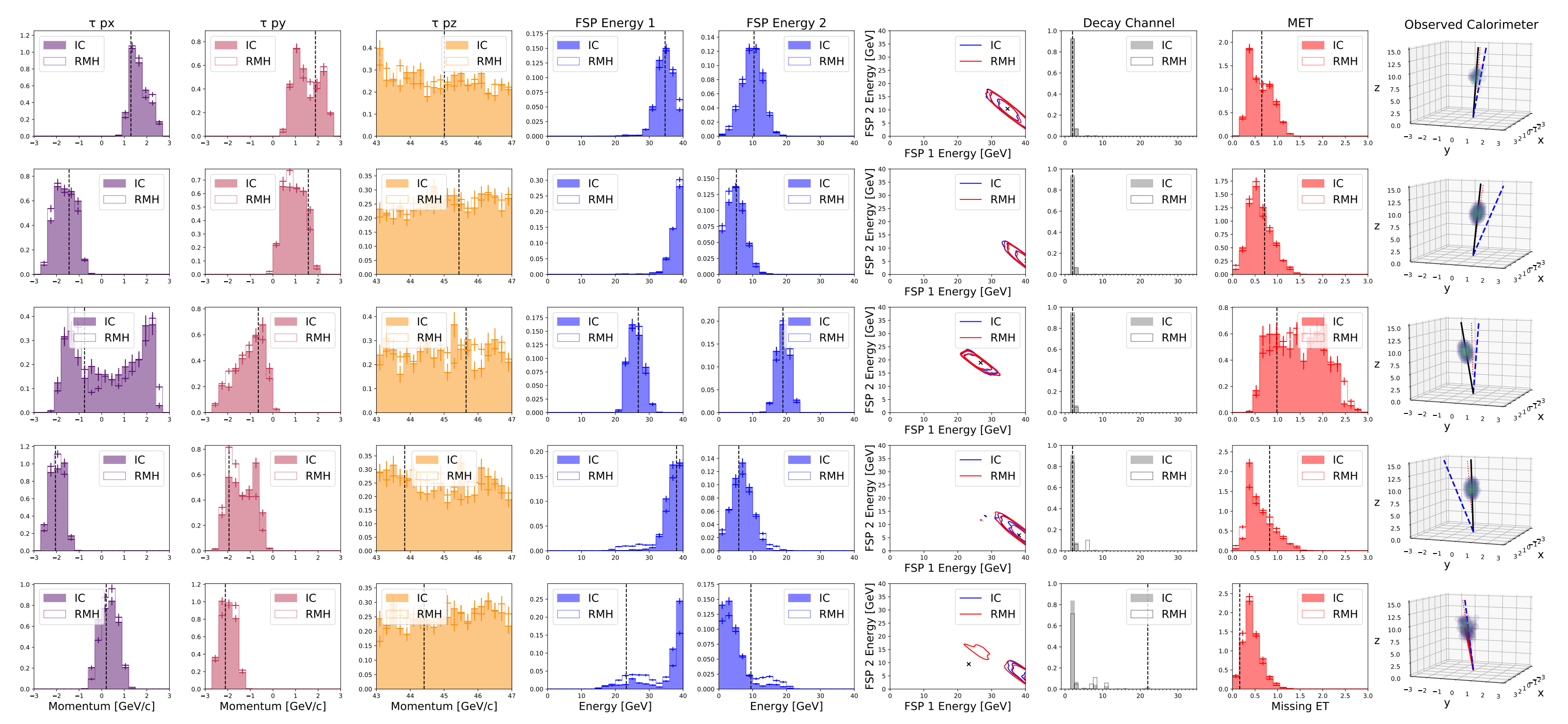
We obtain MCMC baselines in Sherpa's latent space using Random-Walk Metropolis Hastings (RMH), and establish convergence using MCMC diagnostics. This is the first time Bayesian inference in a full particle-physics latent-model (as defined by the Sherpa code base) has been performed. We found the entire latent space to contain at least 25k addresses.[a] Autocorrelation scale is typically around 10^5 iterations.



[a]The simulator defines an unlimited number of latents due to the universal (Turing-complete) nature of the host language and the presence of sampling loops.

Inference Results

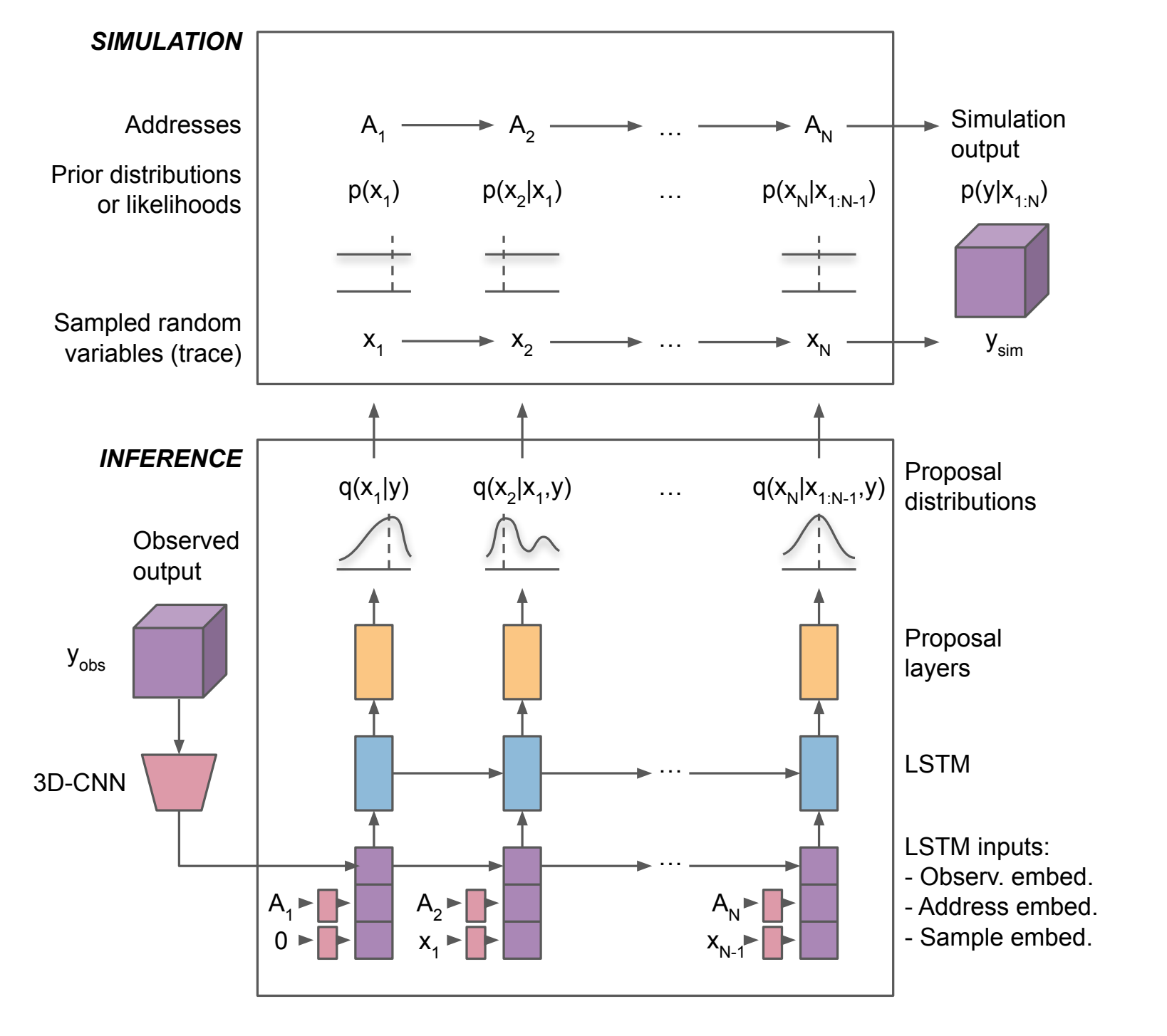
RMH baseline and inference compilation (IC) results where a Channel 2 decay event ($\tau \to \nu_\tau \pi^-$) is the mode of the posterior distribution (a selected subset of latents). RMH and IC took 115 hours and 30 minutes respectively. IC uses importance sampling with learned proposals; it (1) provides non-autocorrelated samples from the posterior, and (2) is embarrassingly parallel.



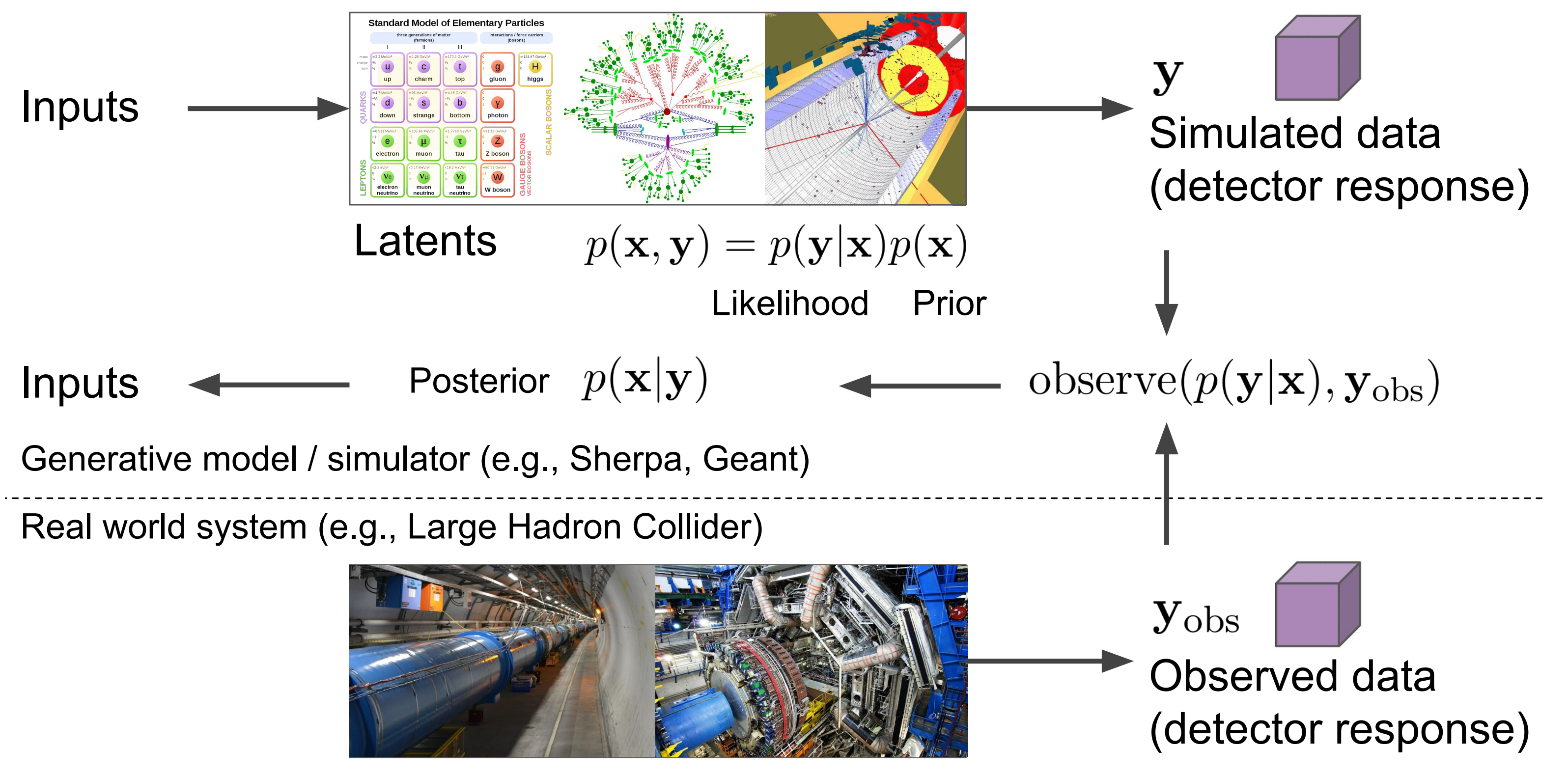
Amortized Inference

We use inference compilation (IC) [4], where an LSTM-based NN is trained with traces sampled from simulator prior $p(\mathbf{x},\mathbf{y})$ to parameterize proposal distributions $q(\mathbf{x}|\mathbf{y})$ in importance sampling. The NN is dynamic: (1) at training, embedding and proposal layers get created as new latents are encountered in executions; (2) at inference, layers are rearranged on-the-fly to match simulator execution.

Sherpa proposal NN has 143M parameters and a 3D-CNN observation embedding for the particle detector.

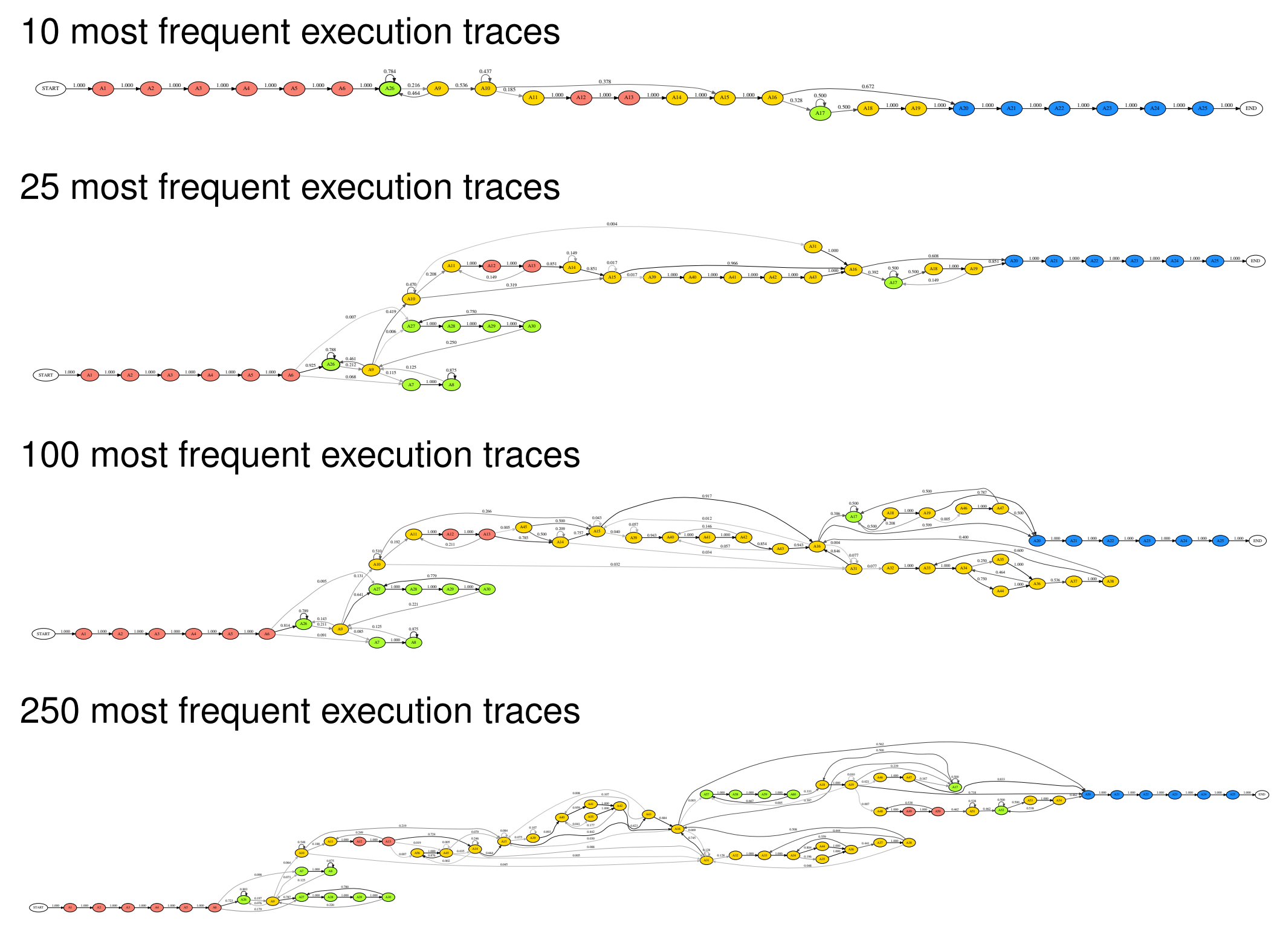


Interpretability


Inputs
Latents
$p(\mathbf{x},\mathbf{y}) = p(\mathbf{y}|\mathbf{x})p(\mathbf{x})$
Likelihood
Prior
$\mathbf{y}$
Simulated data (detector response)
Inputs
Posterior $p(\mathbf{x}|\mathbf{y})$
$\text{observe}(p(\mathbf{y}|\mathbf{x}), \mathbf{y}_{\text{obs}})$
Generative model / simulator (e.g., Sherpa, Geant)
Real world system (e.g., Large Hadron Collider)
$\mathbf{y}_{\text{obs}}$
Observed data (detector response)

We open the black box and discover the latent structure of the probabilistic model implicitly defined by the simulator code base.

All latents can be pinpointed to locations in the code base. Inference results are highly interpretable as we get to see the exact locations and processes in the model that are associated with each prediction.


10 most frequent execution traces
25 most frequent execution traces
100 most frequent execution traces
250 most frequent execution traces

References

[1] Jan-Willem van de Meent, Brooks Paige, Hongseok Yang, and Frank Wood. An introduction to probabilistic programming. *arXiv:1809.10756*, 2018.
[2] Kyle Cranmer, Johann Brehmer, and Gilles Louppe. The frontier of simulation-based inference. *arXiv:1911.01429*, 2019.
[3] Mario Lezcano Casado, Atılım Güneş Baydin, David Martinez Rubio, Tuan Anh Le, Frank Wood, Lukas Heinrich, Gilles Louppe, Kyle Cranmer, Wahid Bhimji, Karen Ng, and Prabhat. Improvements to inference compilation for probabilistic programming in large-scale scientific simulators. In *DLPS Workshop, NeurIPS 2017*, 2017.
[4] Tuan Anh Le, Atılım Güneş Baydin, and Frank Wood. Inference compilation and universal probabilistic programming. In *AISTATS*, 2017.

Paper & Code