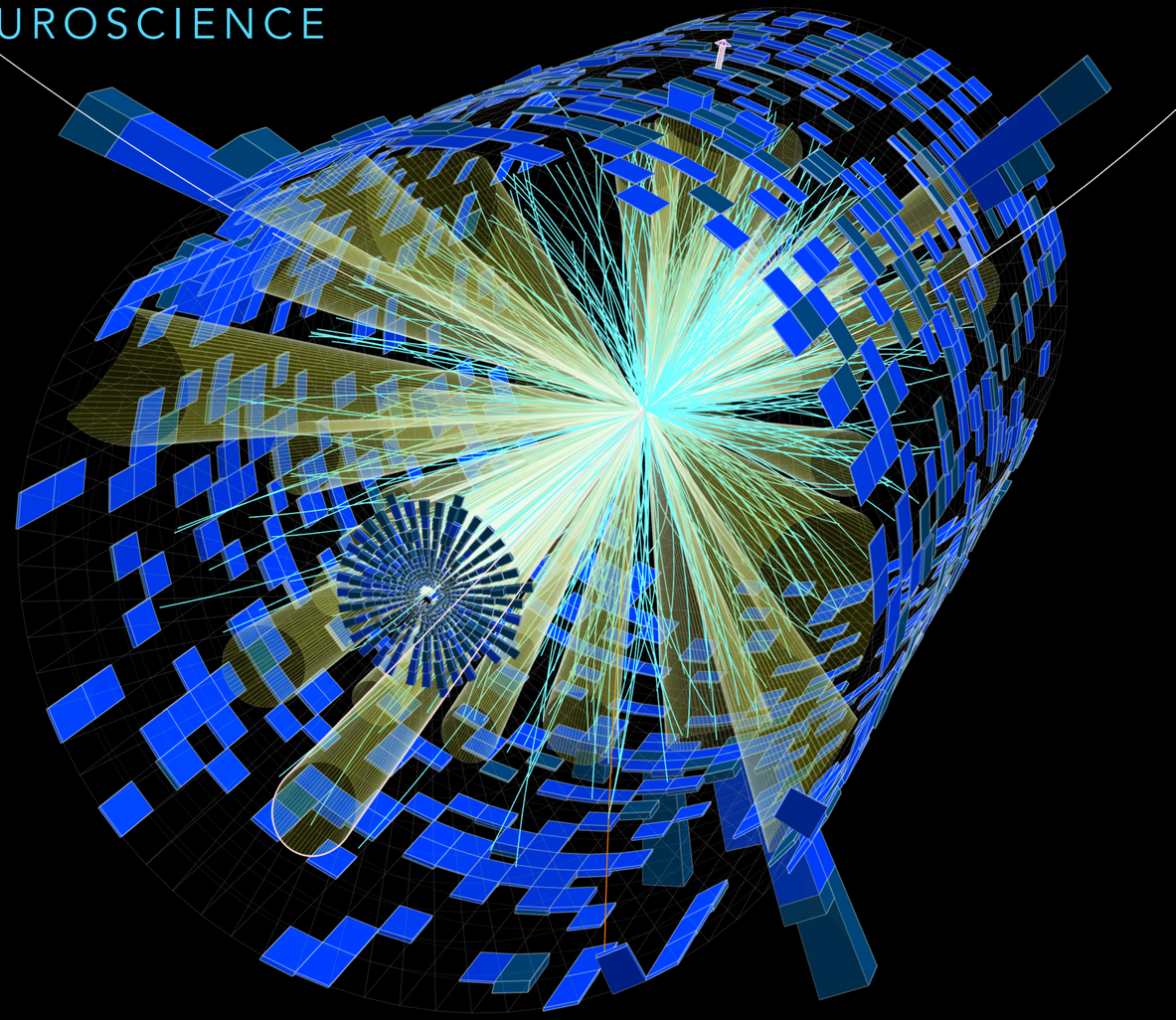
NYU Center for Data Science

Meta AI

Center for Cosmology and Particle Physics

ACCELERATING SIMULATION-BASED INFERENCE

FOR HEP, MM ASTRO, AND COMPUTATIONAL NEUROSCIENCE



@KyleCranmer
New York University
Department of Physics
Center for Data Science
CILVR Lab

Collaborators + Many More

Johann Brehmer
NYU → Qualcom

Gilles Louppe
NYU → U. Liège

Juan Pavez
Santa Maria U.

Lukas Heinrich
NYU → CERN

Atılım Güneş Baydin
U. Oxford

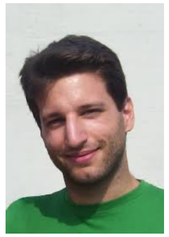
Irina Espejo
NYU

Felix Kling
SLAC

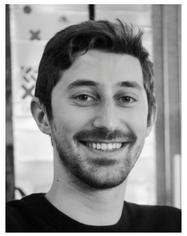
Sebastian Macaluso
NYU

Sid Mishra-Sharma
NYU → IAIFI

Joeri Hermans
NYU → U. Liège



George Papamakarios
DeepMind



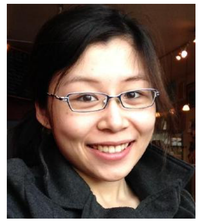
Michael Albergo
NYU

Danilo Rezende
DeepMind

Prabhat
NERSC/ LBNL

Wahid Bhimji
NERSC/ LBNL

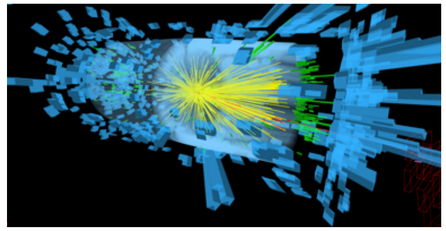
Frank Wood
U. Victoria



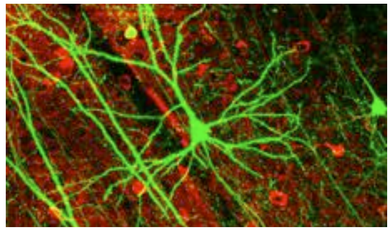
Lei Shao
Intel

Andreas Munk
Oxford

Science is replete with high-fidelity simulators



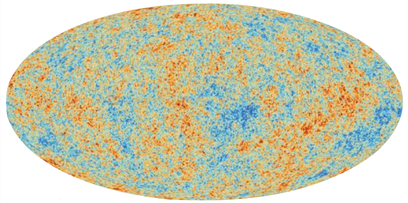
Particle colliders



Neuron activity

Epidemics

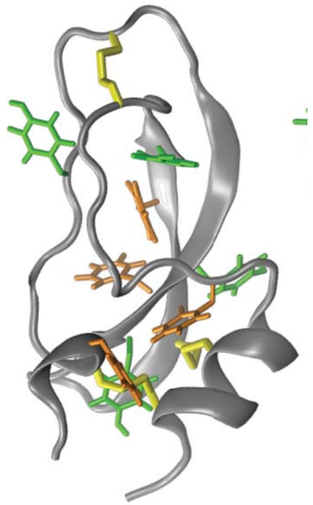
Gravitational lensing

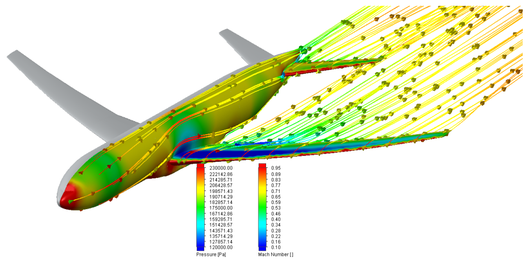


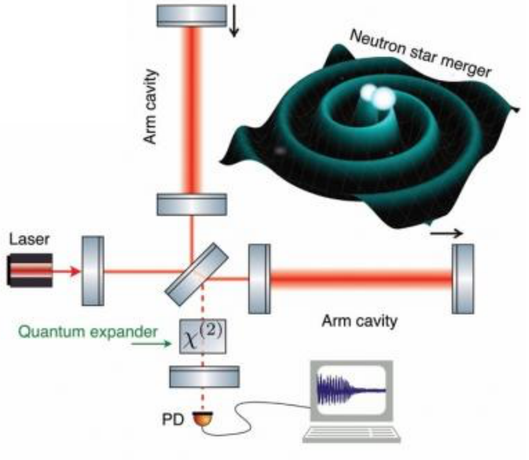
Evolution of the Universe

10^{-18} 10^{-15} 10^{-12} 10^{-9} 10^{-6} 10^{-3} 10^{0} 10^{3} 10^{6} 10^{9} 10^{12} 10^{15} 10^{18} 10^{21} 10^{24} 10^{27}

Length scale [m]



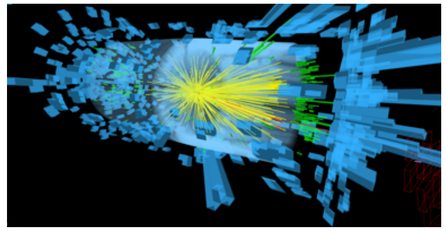




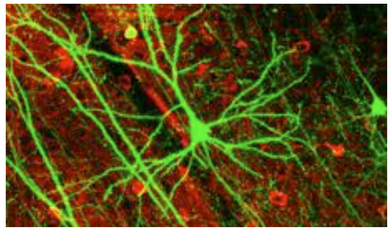
Simulators are causal, generative models of the data generating process

[Cranmer, Brehmer, Louppe PNAS (2020), arXiv:1911.01429]

Science is replete with high-fidelity simulators



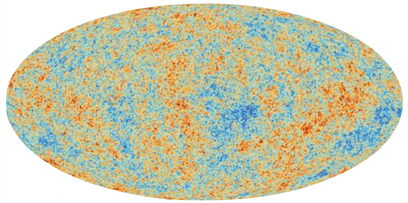
Particle colliders



Neuron activity

Epidemics

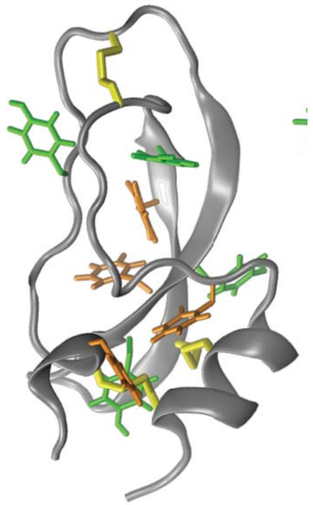
Gravitational lensing

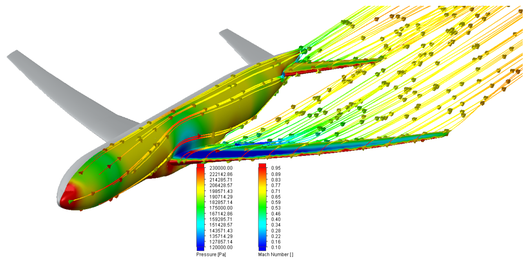


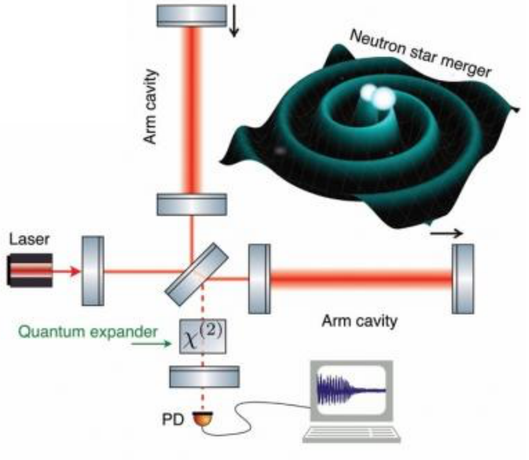
Evolution of the Universe

10^{-18} 10^{-15} 10^{-12} 10^{-9} 10^{-6} 10^{-3} 10^{0} 10^{3} 10^{6} 10^{9} 10^{12} 10^{15} 10^{18} 10^{21} 10^{24} 10^{27}

Length scale [m]



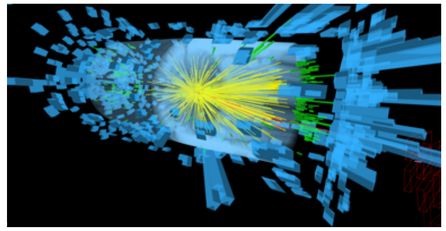




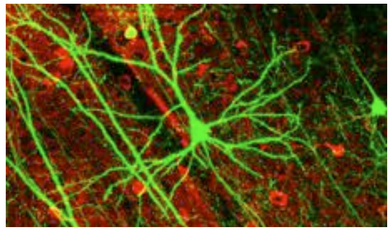
The expressiveness of programming languages facilitates the development of complex, high-fidelity simulations, and the power of modern computing provides the ability to generate synthetic data from them.

[Cranmer, Brehmer, Louppe PNAS (2020), arXiv:1911.01429]

Science is replete with high-fidelity simulators



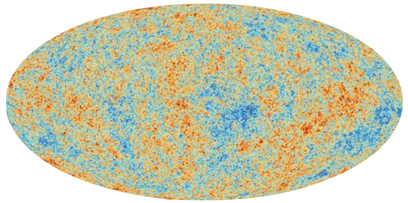
Particle colliders



Neuron activity

Epidemics

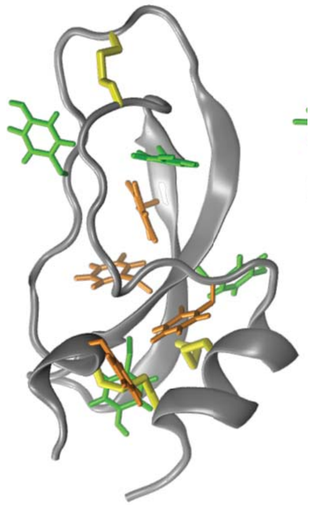
Gravitational lensing

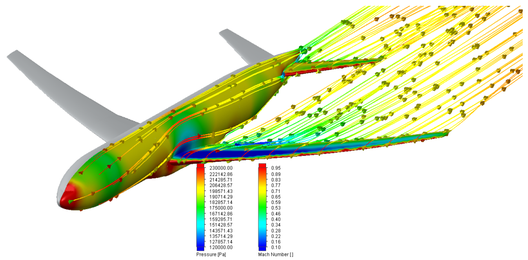


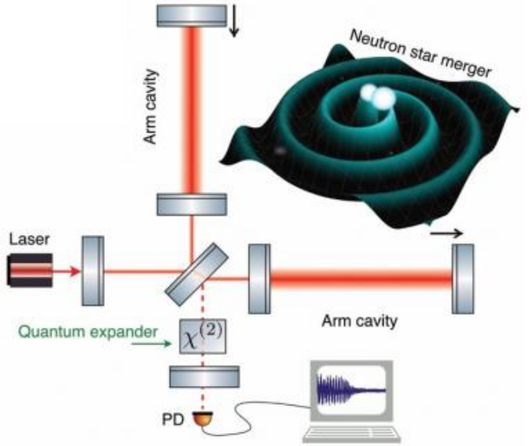
Evolution of the Universe

10^{-18} 10^{-15} 10^{-12} 10^{-9} 10^{-6} 10^{-3} 10^{0} 10^{3} 10^{6} 10^{9} 10^{12} 10^{15} 10^{18} 10^{21} 10^{24} 10^{27}

Length scale [m]



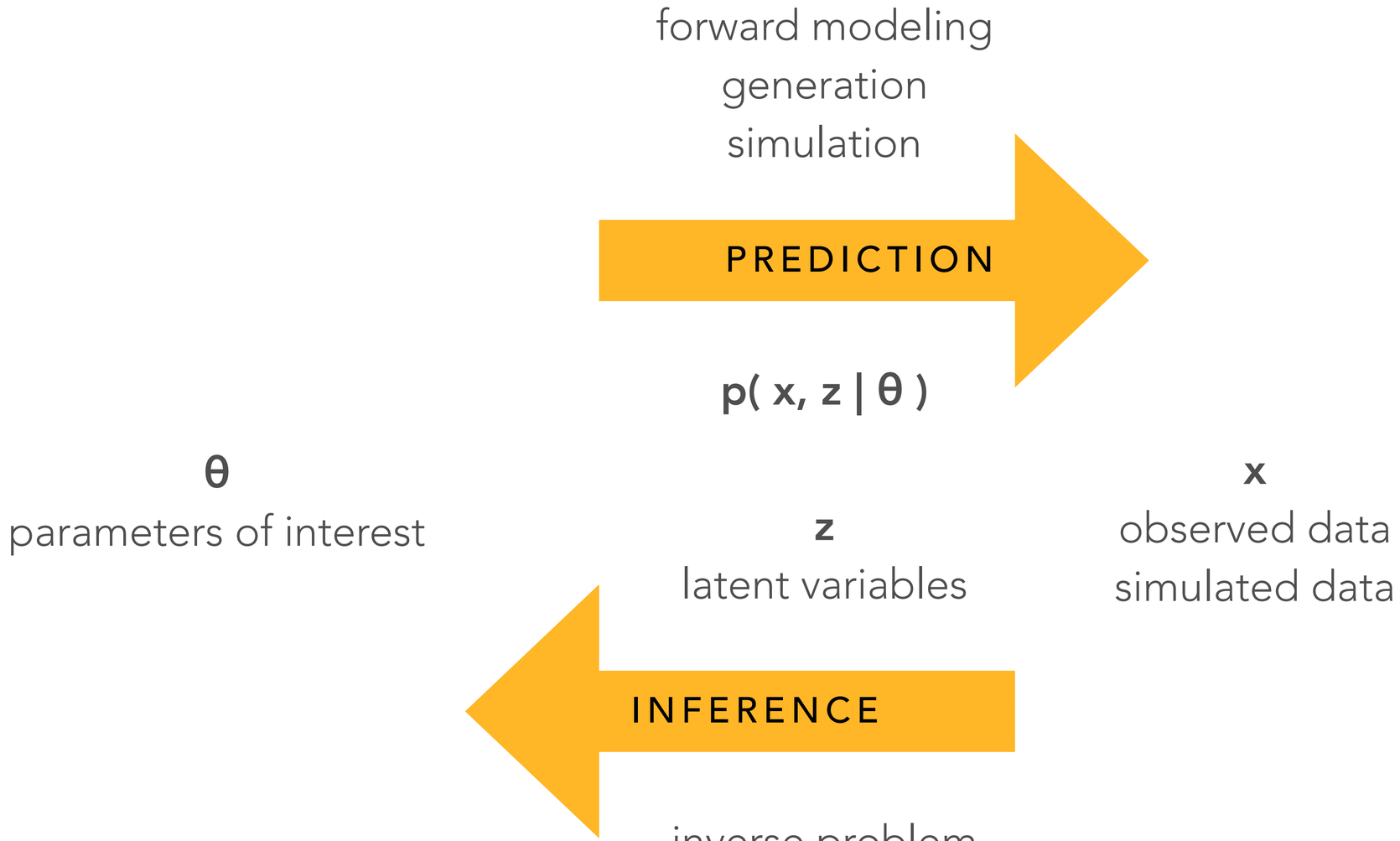




Unfortunately, these simulators are poorly suited for statistical inference.

[Cranmer, Brehmer, Louppe PNAS (2020), arXiv:1911.01429]

Statistical Framing



Model misspecification

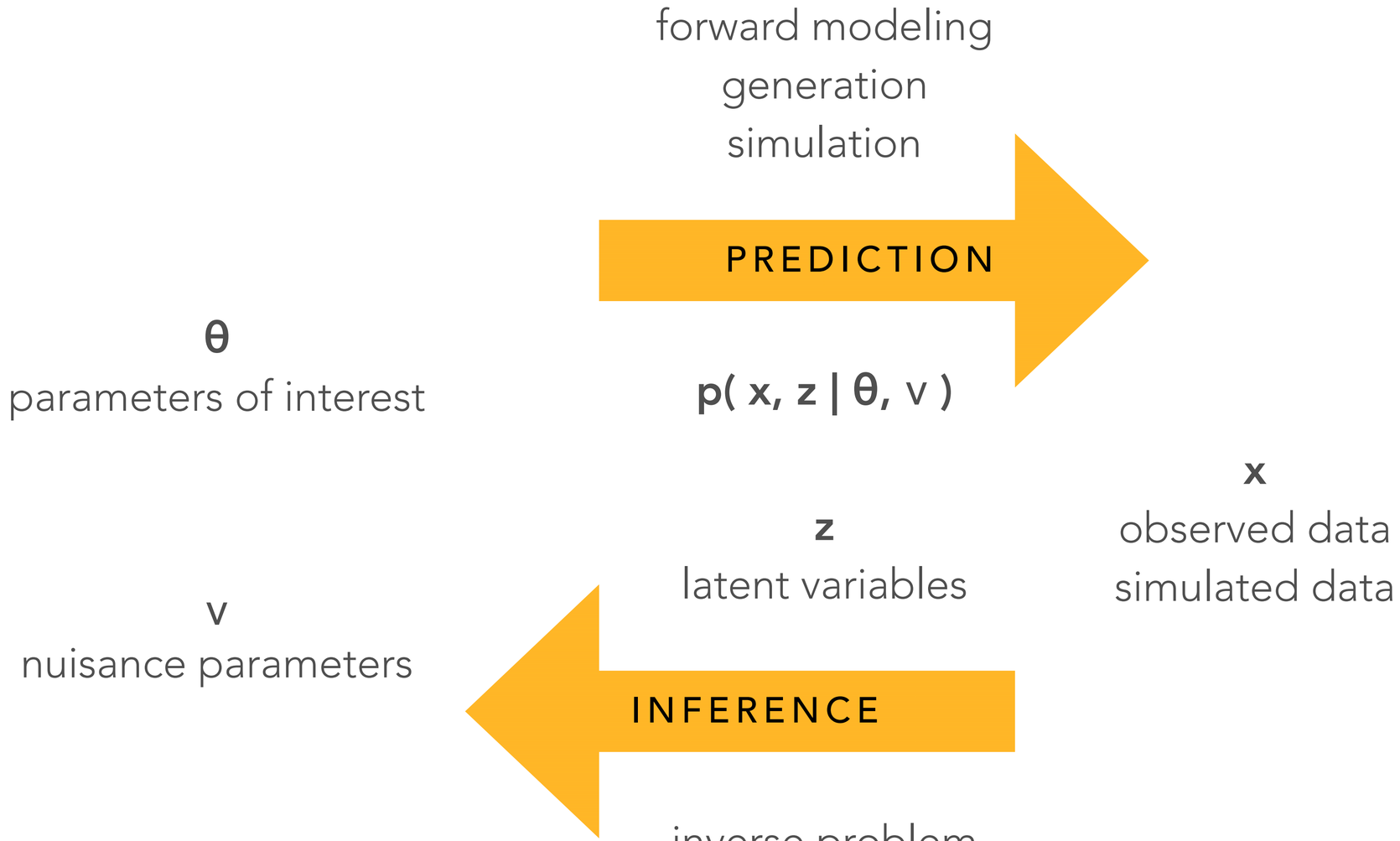
Inference is always done within the context of a model

- If the model is mis-specified it will affect inference
- Here the model **is** the simulator
  - the simulator may not be perfect, but
  - simulators usually include more effects than traditional prescribed models

To account for mis-modeling, simulators are often expanded to model residuals

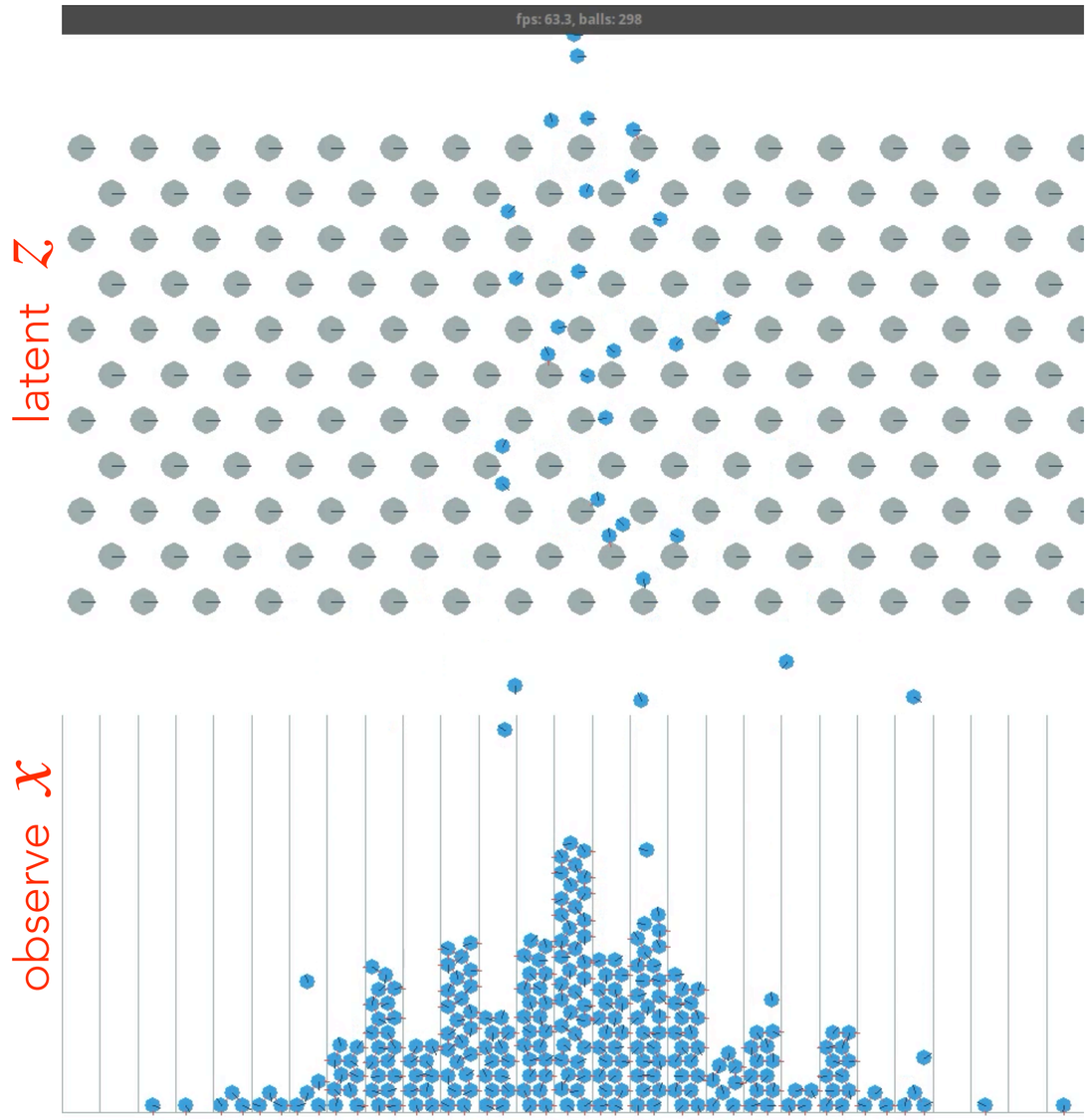
- The simulator now also depends on **nuisance parameters** ν

Statistical Framing



Properties of simulators

ANIMATION BY ATILIM GÜNEŞ BAYDIN



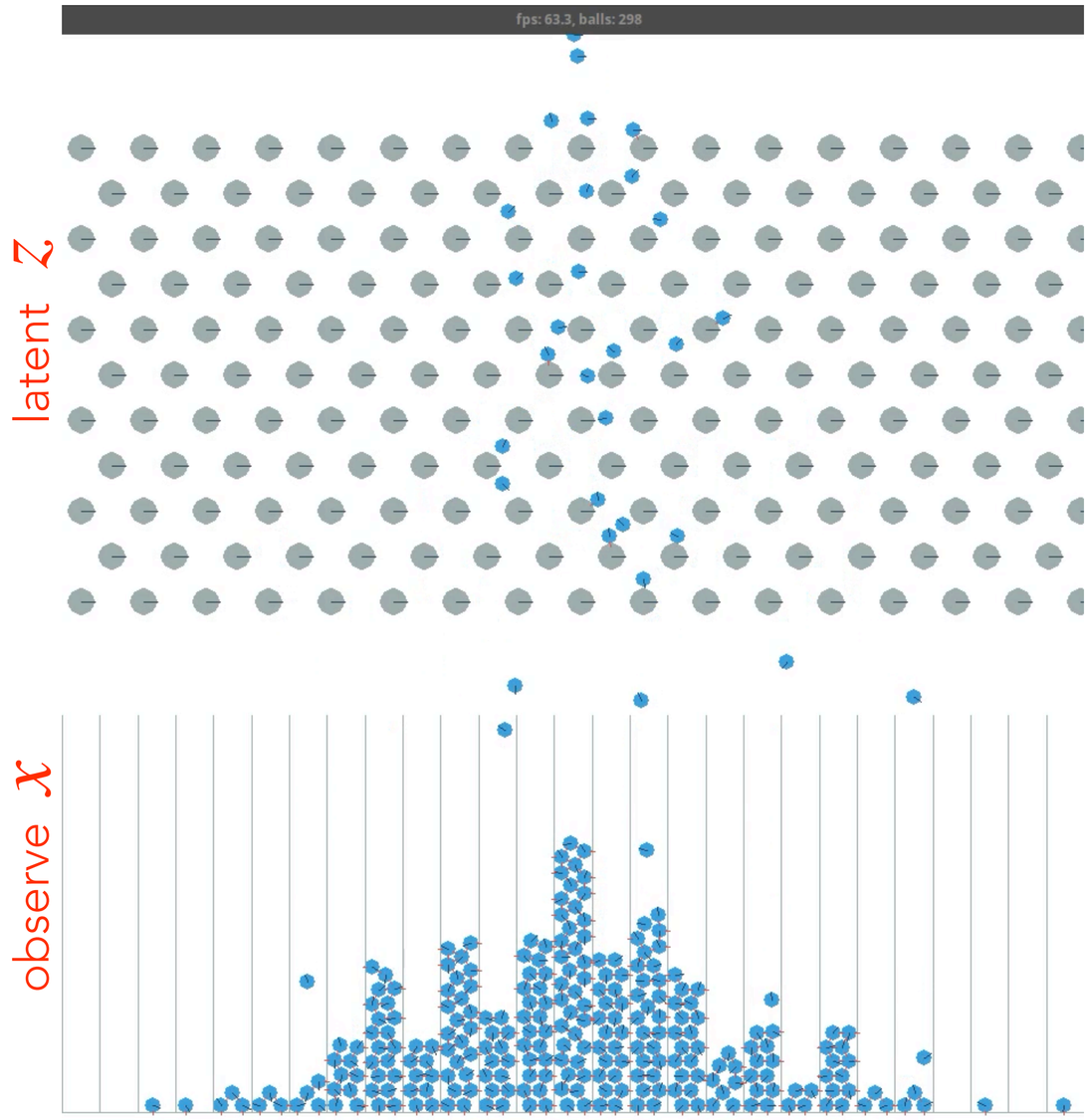
Two broad classes:

- Deterministic evolution of initial state
  - (eg. differential equations, fluid dynamics, N-body simulations, etc.)
- Stochastic evolution
  - (eg. Markov processes, molecular dynamics, Gibbs / Boltzmann distribution in statistical mechanics, stochastic differential equations, etc.)

Integral over latent variables is typically **intractable** $p(x|\theta) = \int p(x, z \mid \theta)\mathrm{d}z$

Properties of simulators

ANIMATION BY ATILIM GÜNEŞ BAYDIN



Two broad classes:

- Deterministic evolution of initial state
  - (eg. differential equations, fluid dynamics, N-body simulations, etc.)
- Stochastic evolution
  - (eg. Markov processes, molecular dynamics, Gibbs / Boltzmann distribution in statistical mechanics, stochastic differential equations, etc.)

Integral over latent variables is typically **intractable** $p(x|\theta) = \int p(x, z \mid \theta)\mathrm{d}z$

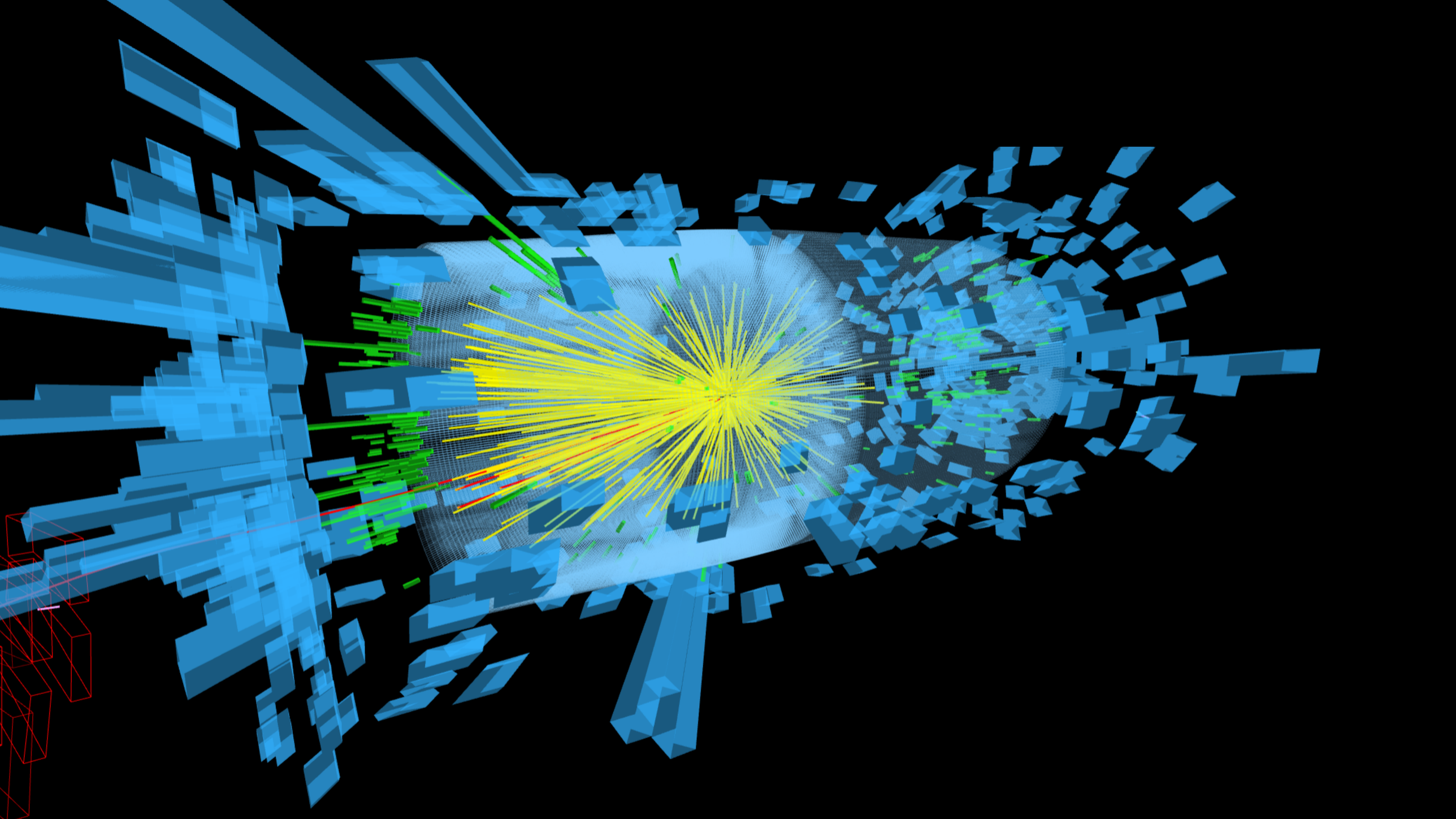
A rose by any other name

This motivates a class of inference methods for a stochastic simulator where

- evaluating the **likelihood is intractable**, but
- it is **possible to sample** synthetic data $x \sim p(x \mid \theta)$

This setting is often referred to as **likelihood-free inference**, but I prefer the term **simulation-based inference** because usually one approximates the likelihood (or likelihood ratio) and then use established inference techniques

- applies to both Bayesian or Frequentist inference



Simulating particle physics processes

$$\mathcal{L}_{SM} = \underbrace{\frac{1}{4}\mathbf{W}_{\mu\nu}\cdot\mathbf{W}^{\mu\nu} - \frac{1}{4}B_{\mu\nu}B^{\mu\nu} - \frac{1}{4}G^{a}_{\mu\nu}G^{\mu\nu}_{a}}_{\text{kinetic energies and self-interactions of the gauge bosons}}$$

$$+ \underbrace{\bar{L}\gamma^{\mu}(i\partial_{\mu} - \frac{1}{2}g\tau\cdot\mathbf{W}_{\mu} - \frac{1}{2}g'YB_{\mu})L \;+\; \bar{R}\gamma^{\mu}(i\partial_{\mu} - \frac{1}{2}g'YB_{\mu})R}_{\text{kinetic energies and electroweak interactions of fermions}}$$

$$+ \underbrace{\frac{1}{2}\left|(i\partial_{\mu} - \frac{1}{2}g\tau\cdot\mathbf{W}_{\mu} - \frac{1}{2}g'YB_{\mu})\,\phi\right|^{2} - V(\phi)}_{W^{\pm},Z,\gamma,\text{and Higgs masses and couplings}}$$

$$+ \underbrace{g''(\bar{q}\gamma^{\mu}T_{a}q)\,G^{a}_{\mu}}_{\text{interactions between quarks and gluons}} + \underbrace{(G_{1}\bar{L}\phi R + G_{2}\bar{L}\phi_{c}R + h.c.)}_{\text{fermion masses and couplings to Higgs}}$$

Simulating particle physics processes

Simulating particle physics processes

Theory
parameters
θ

Evolution

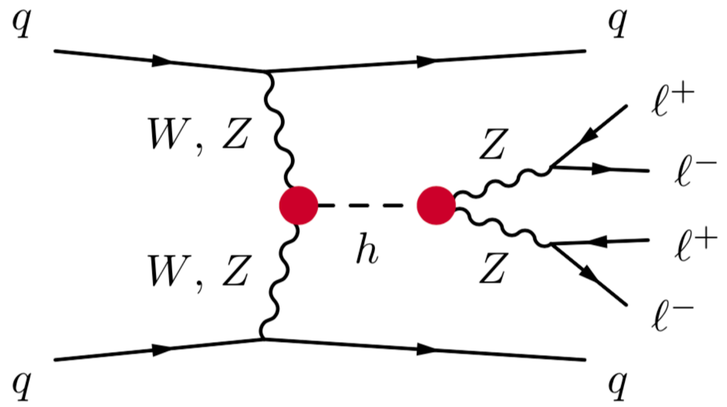
Simulating particle physics processes

Latent variables

Parton-level momenta

Theory parameters

z_p ← θ



Evolution

Simulating particle physics processes

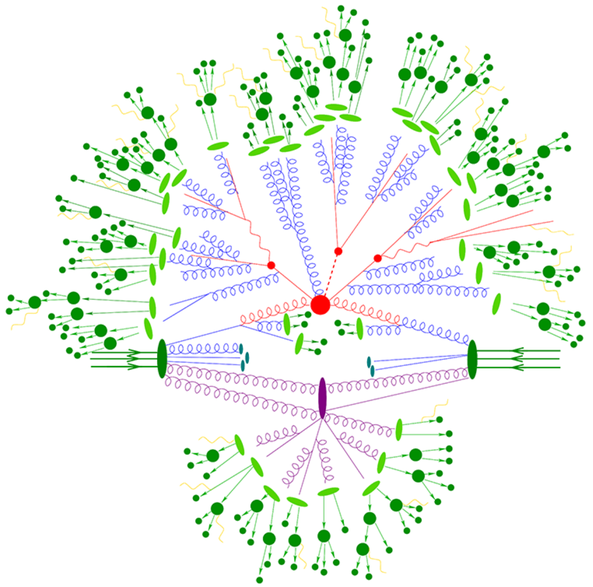
Latent variables

Shower splittings

Parton-level momenta

Theory parameters

z_s ← z_p ← θ



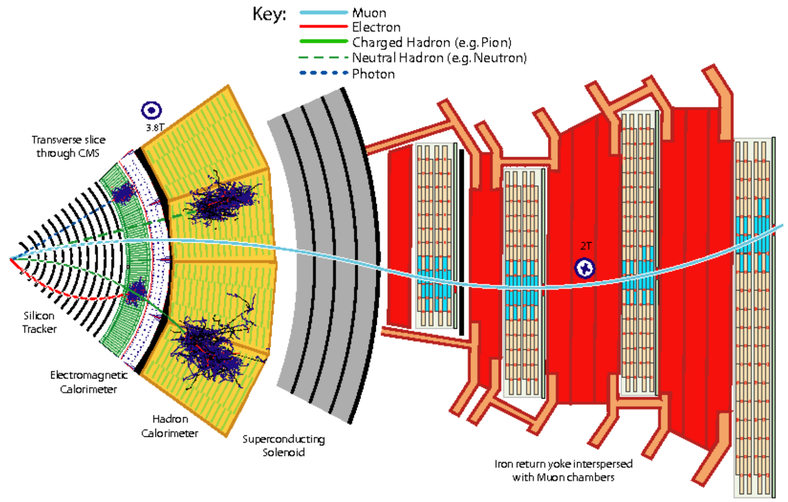
[F. Krauss]

Evolution

Simulating particle physics processes

Latent variables

Detector interactions	Shower splittings	Parton-level momenta	Theory parameters
z_d	z_s	z_p	θ



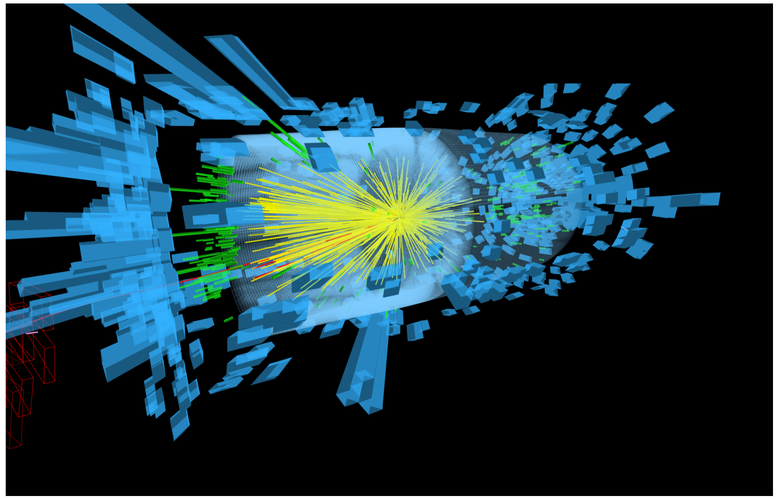
[CMS]

Evolution

Simulating particle physics processes

Latent variables

Observables	Detector interactions	Shower splittings	Parton-level momenta	Theory parameters
x	z_d	z_s	z_p	θ

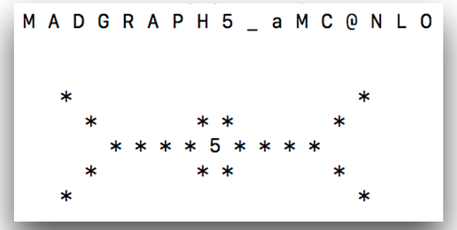


Evolution

Simulating particle physics processes

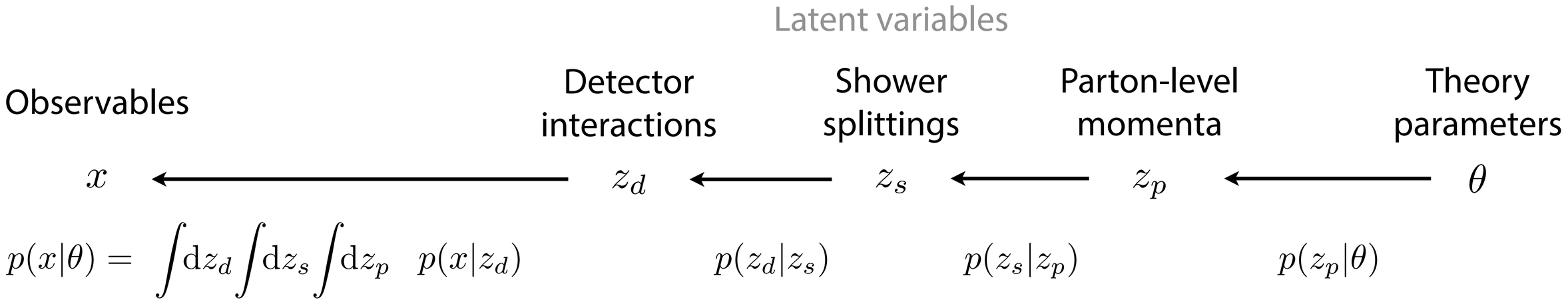
Latent variables

Observables	Detector interactions	Shower splittings	Parton-level momenta	Theory parameters
x	z_d	z_s	z_p	θ
Sample from	$p(x\|z_d)$	$p(z_d\|z_s)$	$p(z_s\|z_p)$	$p(z_p\|\theta)$



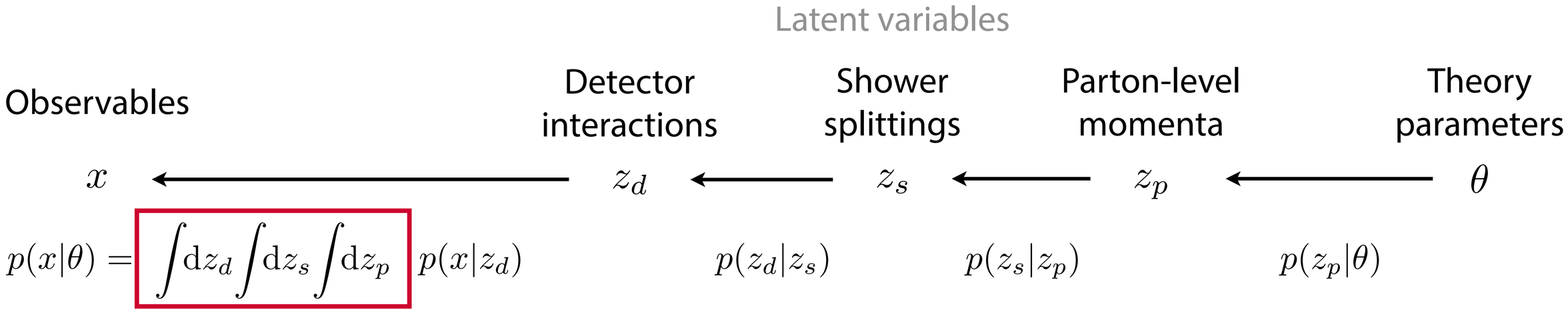
Prediction (simulation)

Simulating particle physics processes



Inference

Simulating particle physics processes



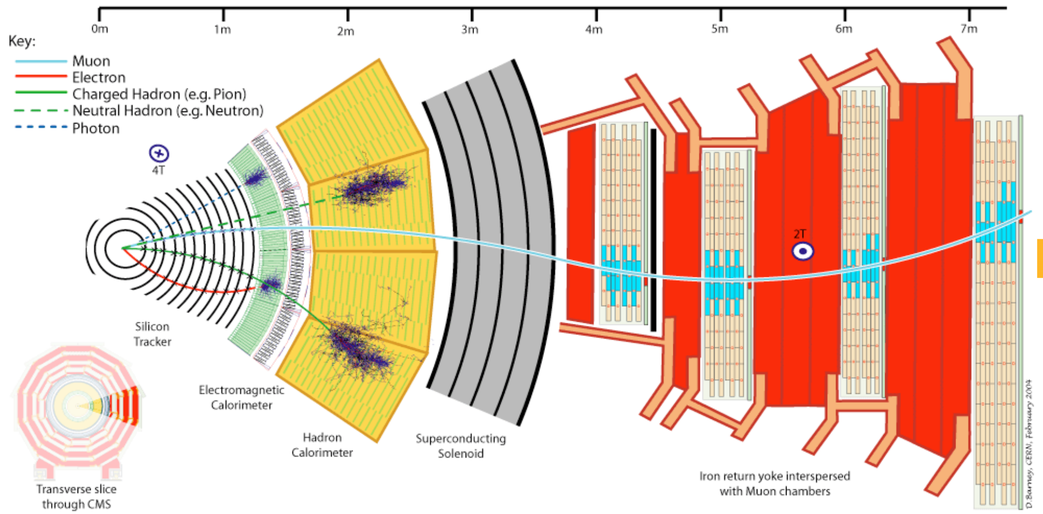
It's infeasible to calculate the integral over this enormous space!

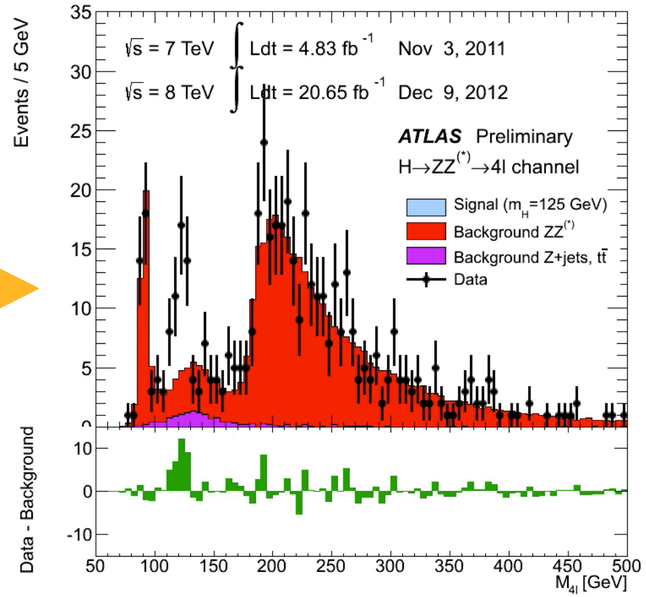
Inference

10^8 sensors → summary statistic

Most measurements and searches for new particles at the LHC are based on the distribution of a single **summary statistic**

- choosing a good summary statistic $s(x)$ (feature engineering) is a task for a skilled physicist and tailored to the goal of measurement or new particle search
- likelihood $p(s \mid \theta)$ **approximated** using histograms or kernel density estimation [Similar to Diggle & Gratton (1984)]

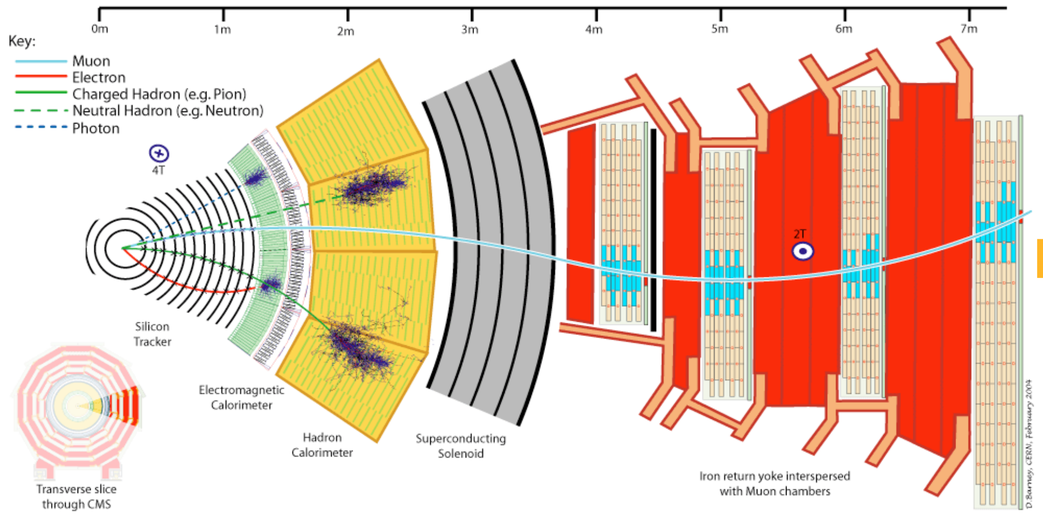


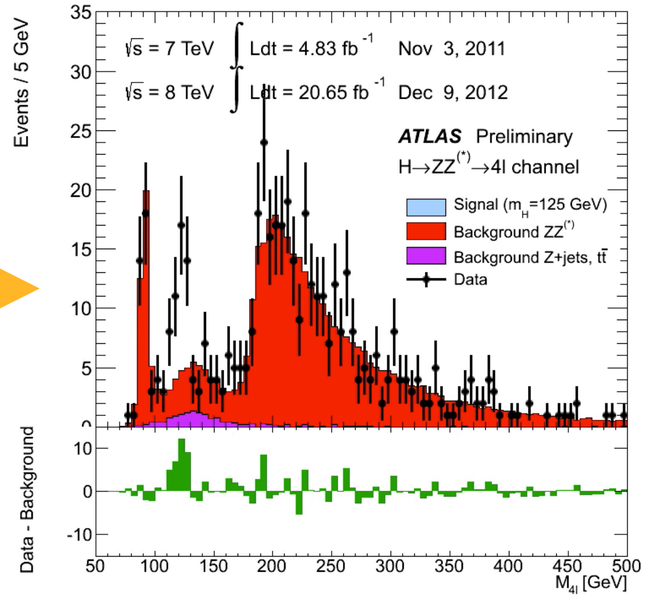


10^8 sensors → summary statistic

Most measurements and searches for new particles at the LHC are based on the distribution of a single **summary statistic**

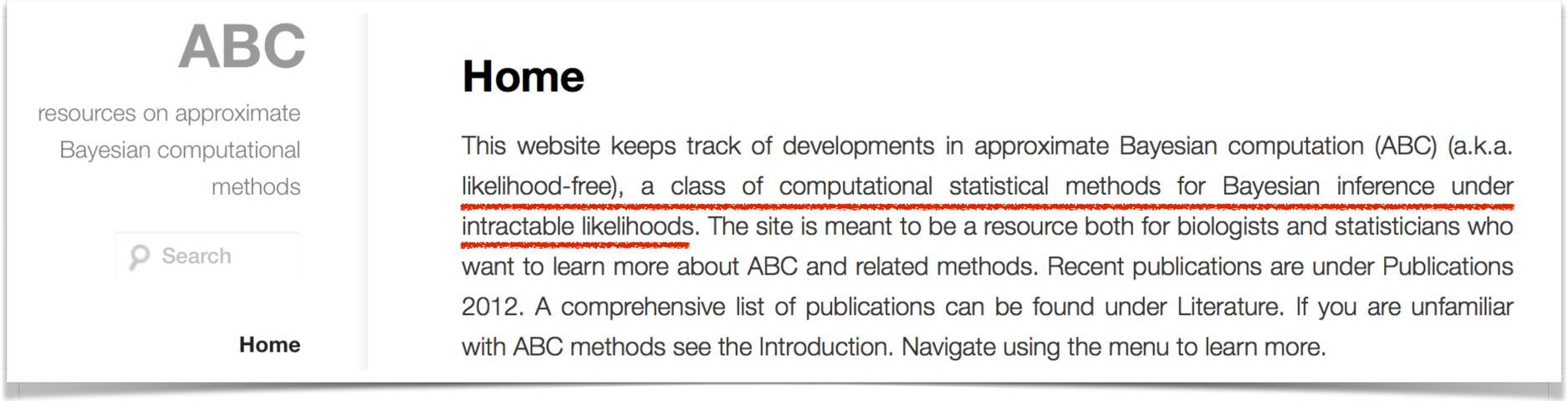
- choosing a good summary statistic $s(x)$ (feature engineering) is a task for a skilled physicist and tailored to the goal of measurement or new particle search
- likelihood $p(s \mid \theta)$ **approximated** using histograms or kernel density estimation [Similar to Diggle & Gratton (1984)]





This doesn't scale if summary is high dimensional!

A common theme, a common language



ABC

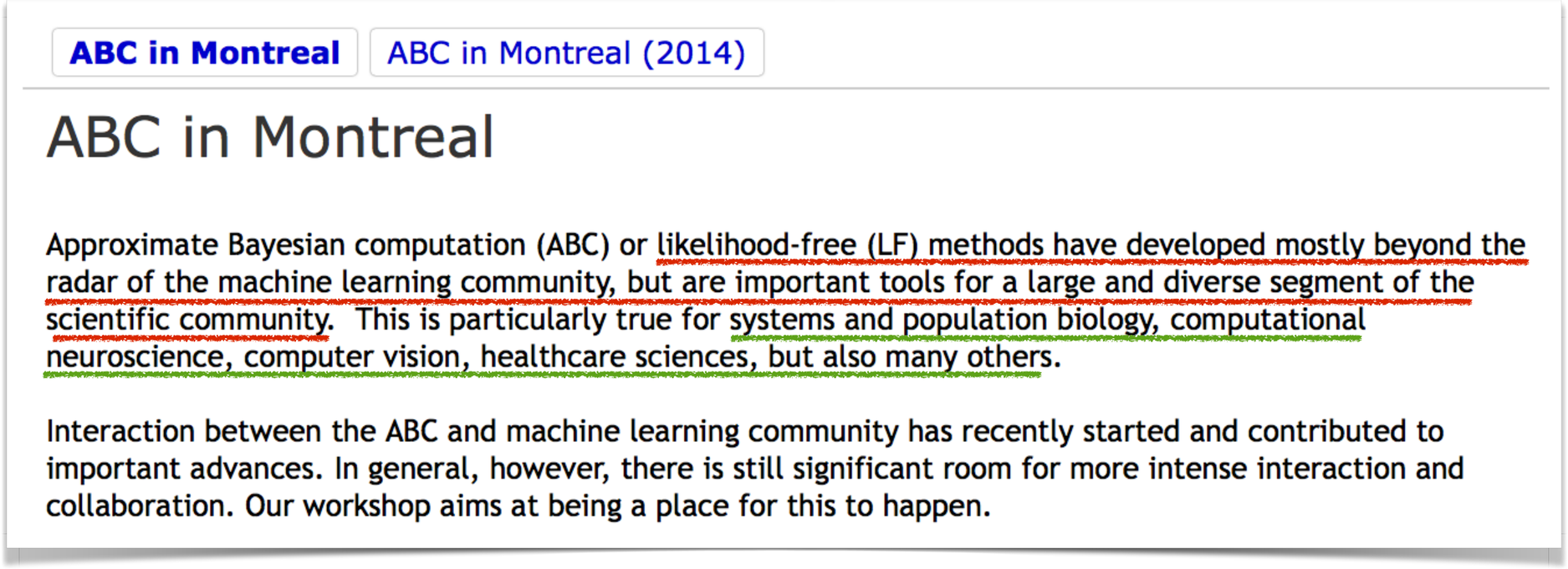
resources on approximate Bayesian computational methods

Search

Home

Home

This website keeps track of developments in approximate Bayesian computation (ABC) (a.k.a. likelihood-free), a class of computational statistical methods for Bayesian inference under intractable likelihoods. The site is meant to be a resource both for biologists and statisticians who want to learn more about ABC and related methods. Recent publications are under Publications 2012. A comprehensive list of publications can be found under Literature. If you are unfamiliar with ABC methods see the Introduction. Navigate using the menu to learn more.



ABC in Montreal | ABC in Montreal (2014)

ABC in Montreal

Approximate Bayesian computation (ABC) or likelihood-free (LF) methods have developed mostly beyond the radar of the machine learning community, but are important tools for a large and diverse segment of the scientific community. This is particularly true for systems and population biology, computational neuroscience, computer vision, healthcare sciences, but also many others.

Interaction between the ABC and machine learning community has recently started and contributed to important advances. In general, however, there is still significant room for more intense interaction and collaboration. Our workshop aims at being a place for this to happen.

Markov chain Monte Carlo without likelihoods

Paul Marjoram*, John Molitor*, Vincent Plagnol†, and Simon Tavaré†‡

*Biostatistics Division, Department of Preventive Medicine, Keck School of Medicine, and †Molecular and Computational Biology, Department of Biological Sciences, University of Southern California, Los Angeles, CA 90089

D1. Generate θ from $\pi(\cdot)$.
D2. Simulate $\mathcal{D}'$ from stochastic model $\mathcal{M}$ with parameter θ, and compute the corresponding statistics S'.
D3. Calculate the distance $\rho(S, S')$ between S and S'.
D4. Accept θ if $\rho \leq \varepsilon$, and return to *D1*.

discussion.

One of the basic problems in Bayesian statistics is the computation of posterior distributions. We imagine data $\mathcal{D}$ generated from a model $\mathcal{M}$ determined by parameters θ, the prior density of which is denoted by $\pi(\theta)$. We assume, unless otherwise stated that the data are discrete. The posterior distribution of interest is $f(\theta|\mathcal{D})$, which is given by

$$f(\theta|\mathcal{D}) = \mathbb{P}(\mathcal{D}|\theta)\pi(\theta)/\mathbb{P}(\mathcal{D}) \qquad [1]$$

where $\mathbb{P}(\mathcal{D}) = \int \mathbb{P}(\mathcal{D}|\theta)\pi(\theta)d\theta$ is the normalizing constant.

In most scientific contexts, explicit formulae for such posterior densities are few and far between, and we usually resort to stochastic simulation to generate observations from f. Perhaps the simplest approach for this is the rejection method:

A1. Generate θ from $\pi(\cdot)$.
A2. Accept θ with probability $h = \mathbb{P}(\mathcal{D}|\theta)$; return to *A1*.

typically larger than $\mathbb{P}(\mathcal{D})$, resulting in more acceptances. In practice it will be hard, if not impossible, to identity a suitable set of sufficient statistics, and we then might resort to a more heuristic approach. Thus we seek to use knowledge of the particular problem at hand to suggest summary statistics that capture information about θ. With these statistics in hand, we have the following approximate Bayesian computation scheme for data $\mathcal{D}$ summarized by S:

D1. Generate θ from $\pi(\cdot)$.
D2. Simulate $\mathcal{D}'$ from stochastic model $\mathcal{M}$ with parameter θ, and compute the corresponding statistics S'.
D3. Calculate the distance $\rho(S, S')$ between S and S'.
D4. Accept θ if $\rho \leq \varepsilon$, and return to *D1*.

There are several advantages to these rejection methods, among them the fact that they are usually easy to code, they generate independent observations (and thus can use embarrassingly parallel computation), and they readily provide estimates of Bayes factors that can be used for model com-

Markov chain Monte Carlo without likelihoods

Paul Marjoram*, John Molitor*, Vincent Plagnol†, and Simon Tavaré†‡

*Biostatistics Division, Department of Preventive Medicine, Keck School of Medicine, and †Molecular and Computational Biology, Department of Biological Sciences, University of Southern California, Los Angeles, CA 90089

Communicated by Michael S. Waterman, University of Southern California, Los Angeles, CA, October 24, 2003 (received for review June 20, 2003)

Many stochastic simulation approaches for generating observations from a posterior distribution depend on knowing a likelihood function. However, for many complex probability models, such likelihoods are either impossible or computationally prohibitive to obtain. Here we present a Markov chain Monte Carlo method for generating observations from a posterior distribution without the use of likelihoods. It can also be used in frequentist applications, in particular for maximum-likelihood estimation. The approach is illustrated by an example of ancestral inference in population genetics. A number of open problems are highlighted in the discussion.

One of the basic problems in Bayesian statistics is the computation of posterior distributions. We imagine data $\mathcal{D}$ generated from a model $\mathcal{M}$ determined by parameters θ, the prior density of which is denoted by $\pi(\theta)$. We assume unless otherwise stated that the data are discrete. The posterior distribution of interest is $f(\theta|\mathcal{D})$, which is given by

A1. Generate θ from $\pi(\cdot)$.
A2. Accept θ with probability $h = \mathbb{P}(\mathcal{D}|\theta)$; return to *A1*.

of ε therefore reflects a tension between computability and accuracy. The method is still honest in that, for a given ρ and ε, we are generating independent and identically distributed observations from $f(\theta|\rho(\mathcal{D}, \mathcal{D}') \leq \varepsilon)$.

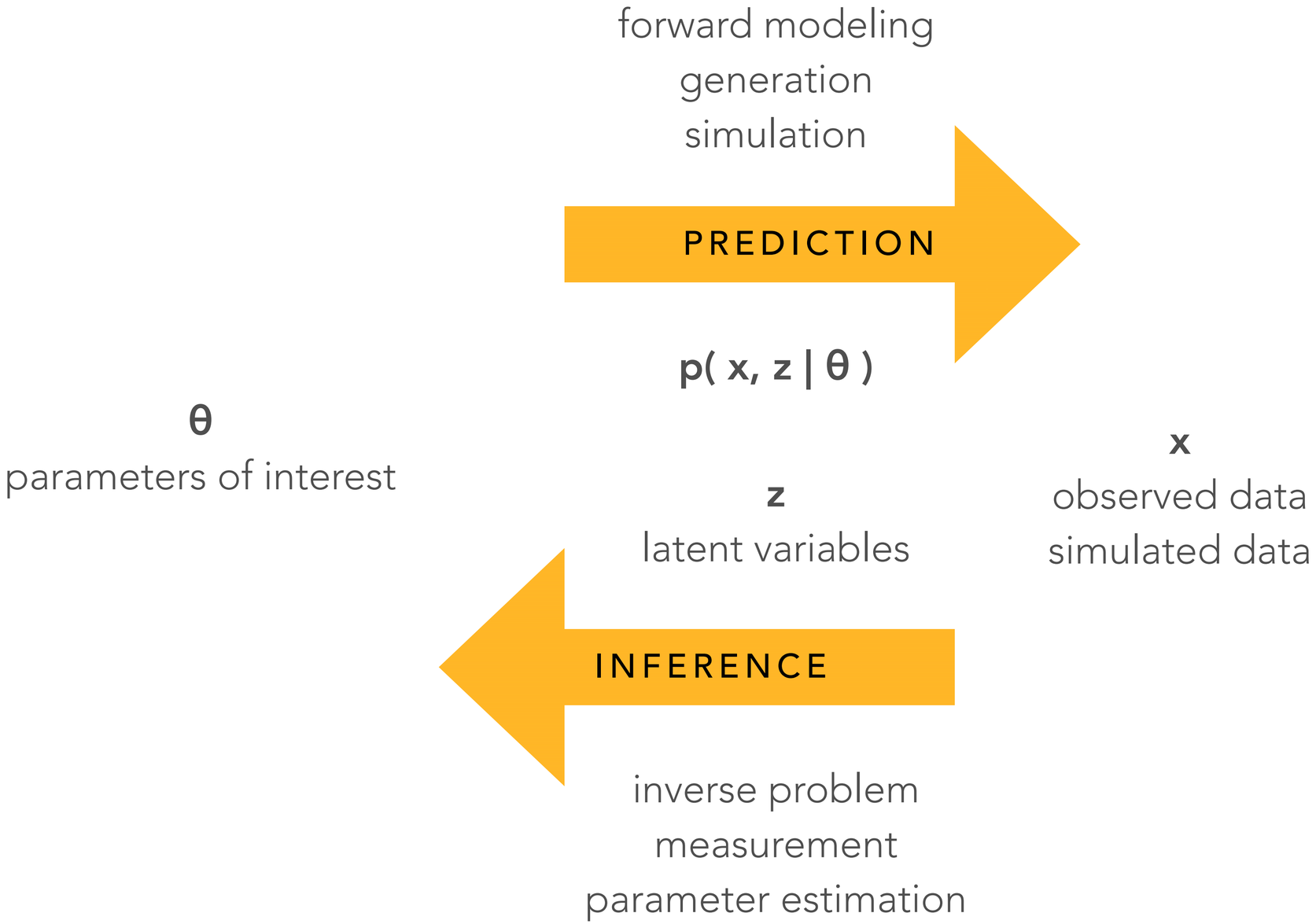
When $\mathcal{D}$ is high-dimensional or continuous, this approach can be impractical as well, and then the comparison of $\mathcal{D}'$ with $\mathcal{D}$ can be made by using lower-dimensional summaries of the data. The motivation for this approach is that if the set of statistics $S = (S_1, \ldots, S_p)$ is sufficient for θ, in that $\mathbb{P}(\mathcal{D}|S, \theta)$ is independent of θ, then $f(\theta|\mathcal{D}) = f(\theta|S)$. The normalizing constant $\mathbb{P}(S)$ is typically larger than $\mathbb{P}(\mathcal{D})$, resulting in more acceptances. In practice it will be hard, if not impossible, to identity a suitable set of sufficient statistics, and we then might resort to a more heuristic approach. Thus we seek to use knowledge of the particular problem at hand to suggest summary statistics that capture information about θ. With these statistics in hand, we have the following approximate Bayesian computation scheme for data $\mathcal{D}$ summarized by S:

practice it will be hard, if not impossible, to identity a suitable set of sufficient statistics, and we then might resort to a more heuristic approach.

θ, and

nods,
, they
generate independent observations (and thus can use embarrassingly parallel computation), and they readily provide estimates of Bayes factors that can be used for model com-

Statistical Framing



Statistical Framing

forward modeling
generation
simulation

PREDICTION

$p(x, z \mid \theta)$

θ
parameters of interest

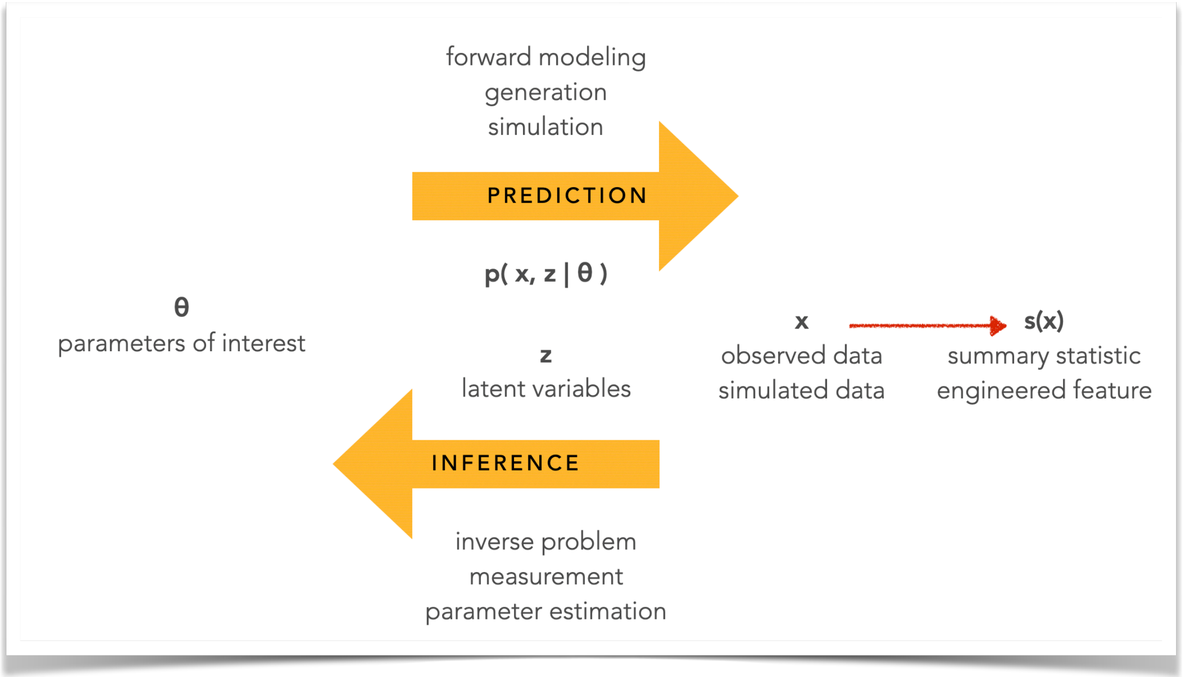
z
latent variables

x → $s(x)$
observed data → summary statistic
simulated data → engineered feature

INFERENCE

inverse problem
measurement
parameter estimation

Forward modeling and inverse problems



$s \in \mathbb{R}^o$

summary statistic

$x \in \mathrm{X}$

observed data

$z \in \mathrm{Z}$

latent variables

$\theta \in R^p$

parameters of interest

1 10 100 1,000 10,000 100,000

Dimensionality

A review

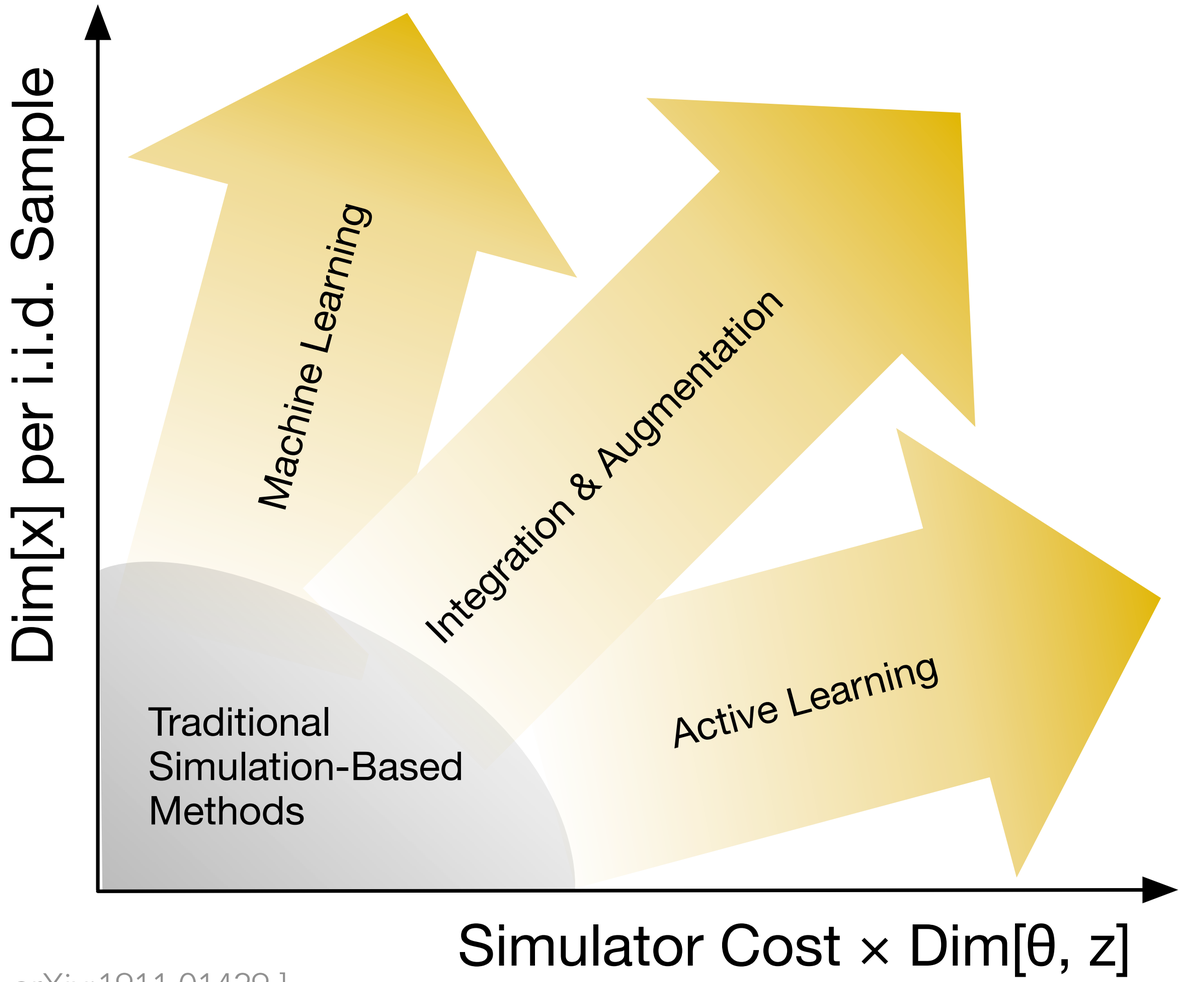
Published in Proceedings of the National Academy of Sciences

The frontier of simulation-based inference

Kyle Cranmer[a,b,1], **Johann Brehmer**[a,b], **and Gilles Louppe**[c]

[a]Center for Cosmology and Particle Physics, New York University, USA; [b]Center for Data Science, New York University, USA; [c]Montefiore Institute, University of Liège, Belgium

April 3, 2020



Gilles Louppe

Johann Brehmer

[Cranmer, Brehmer, Louppe PNAS (2020), arXiv:1911.01429]

ICML 2017 Workshop on Implicit Models

Workshop Aims

Probabilistic models are an important tool in machine learning. They form the basis for models that generate realistic data, uncover hidden structure, and make predictions. Traditionally, probabilistic models in machine learning have focused on prescribed models. Prescribed models specify a joint density over observed and hidden variables that can be easily evaluated. The requirement of a tractable density simplifies their learning but limits their flexibility --- several real world phenomena are better described by simulators that do not admit a tractable density. Probabilistic models defined only via the simulations they produce are called implicit models.

Arguably starting with generative adversarial networks, research on implicit models in machine learning has exploded in recent years. This workshop's aim is to foster a discussion around the recent developments and future directions of implicit models.

Implicit models have many applications. They are used in ecology where models simulate animal populations over time; they are used in phylogeny, where simulations produce hypothetical ancestry trees; they are used in physics to generate particle simulations for high energy processes. Recently, implicit models have been used to improve the state-of-the-art in image and content generation. Part of the workshop's focus is to discuss the commonalities among applications of implicit models.

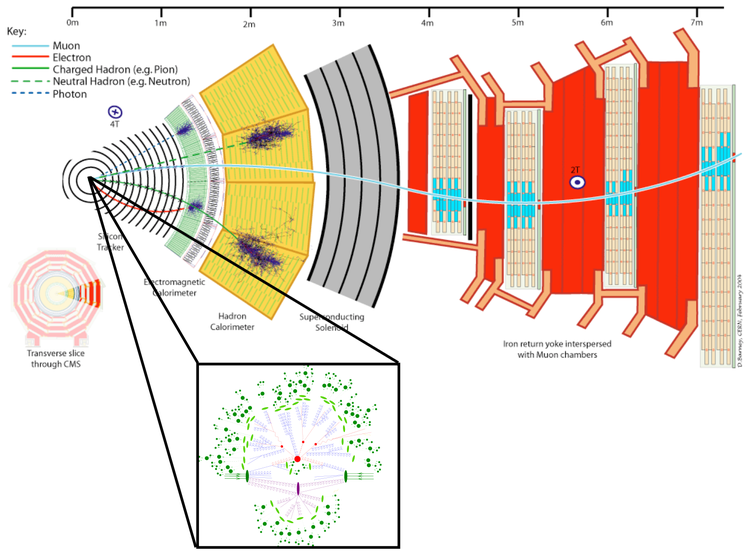
Of particular interest at this workshop is to unite fields that work on implicit models. For example:

- **Generative adversarial networks** (a NIPS 2016 workshop) are implicit models with an adversarial training scheme.
- Recent advances in **variational inference** (a NIPS 2015 and 2016 workshop) have leveraged implicit models for more accurate approximations.
- **Approximate Bayesian computation** (a NIPS 2015 workshop) focuses on posterior inference for models with implicit likelihoods.
- Learning implicit models is deeply connected to **two sample testing, density ratio and density difference** estimation.

We hope to bring together these different views on implicit models, identifying their core challenges and combining their innovations.

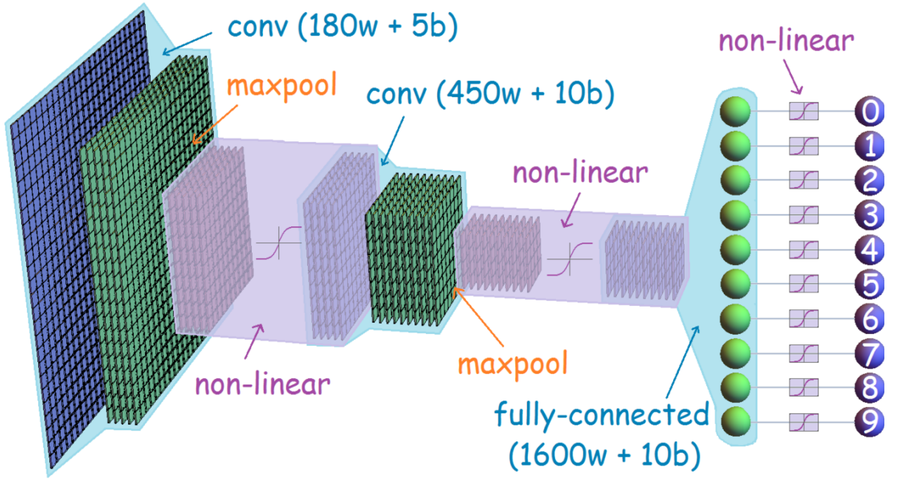
Two approaches simulation-based inference

Use simulator
(much more efficiently)



- Approximate Bayesian Computation (ABC)
- Probabilistic Programming
- Adversarial Variational Optimization

Learn simulator
(with deep learning)



- Likelihood ratio trick (with classifiers)
- Conditional density estimate (with normalizing flows)
- Learned summary statistics

[image credit: A.P. Goucher]

From the review

[Cranmer, J. Brehmer, G. Louppe, PNAS (2020), arXiv:1911.01429]

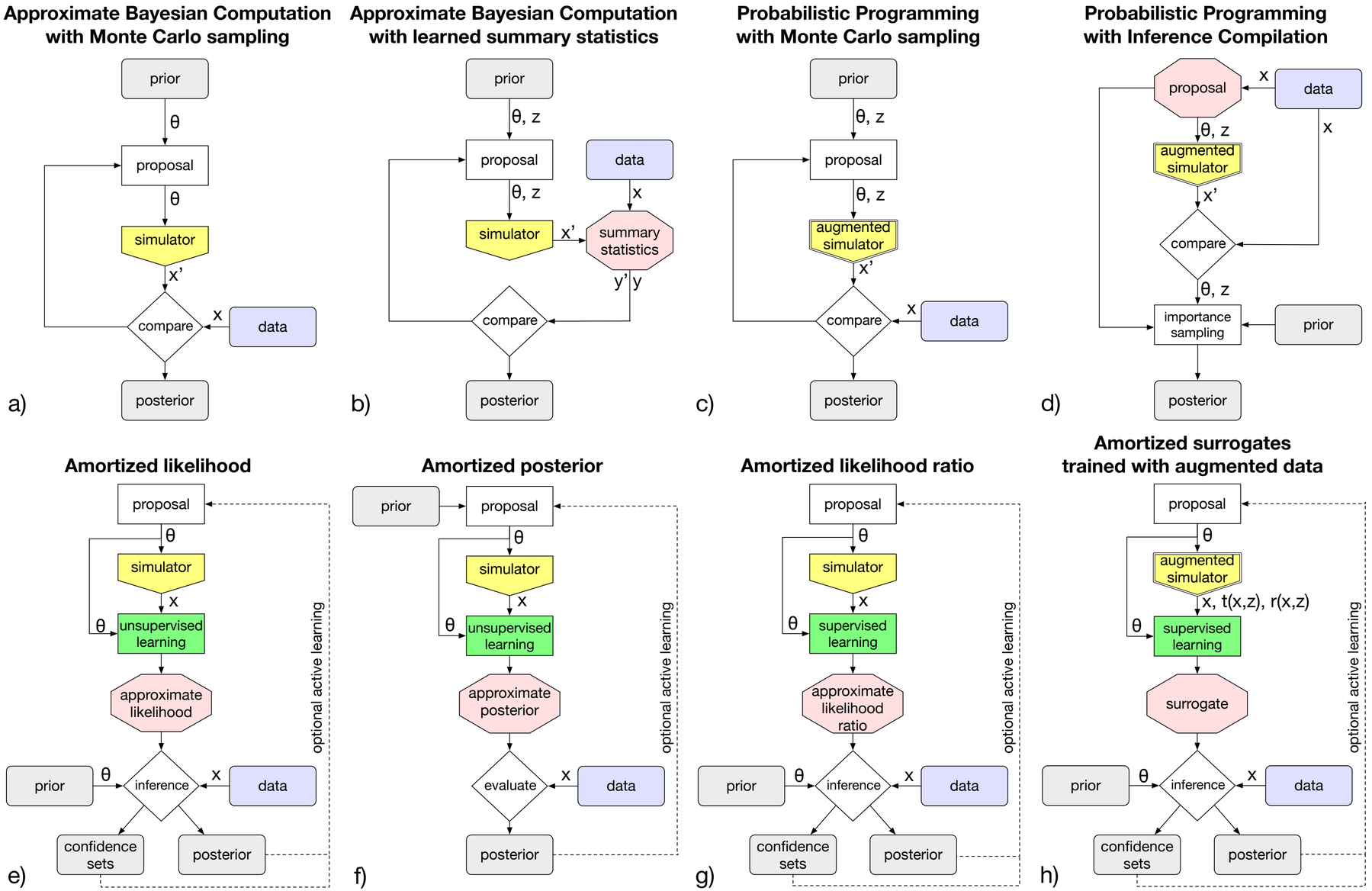


Fig. 3. Overview of different approaches to simulation-based inference.

From the review

[Cranmer, J. Brehmer, G. Louppe, PNAS (2020), arXiv:1911.01429]

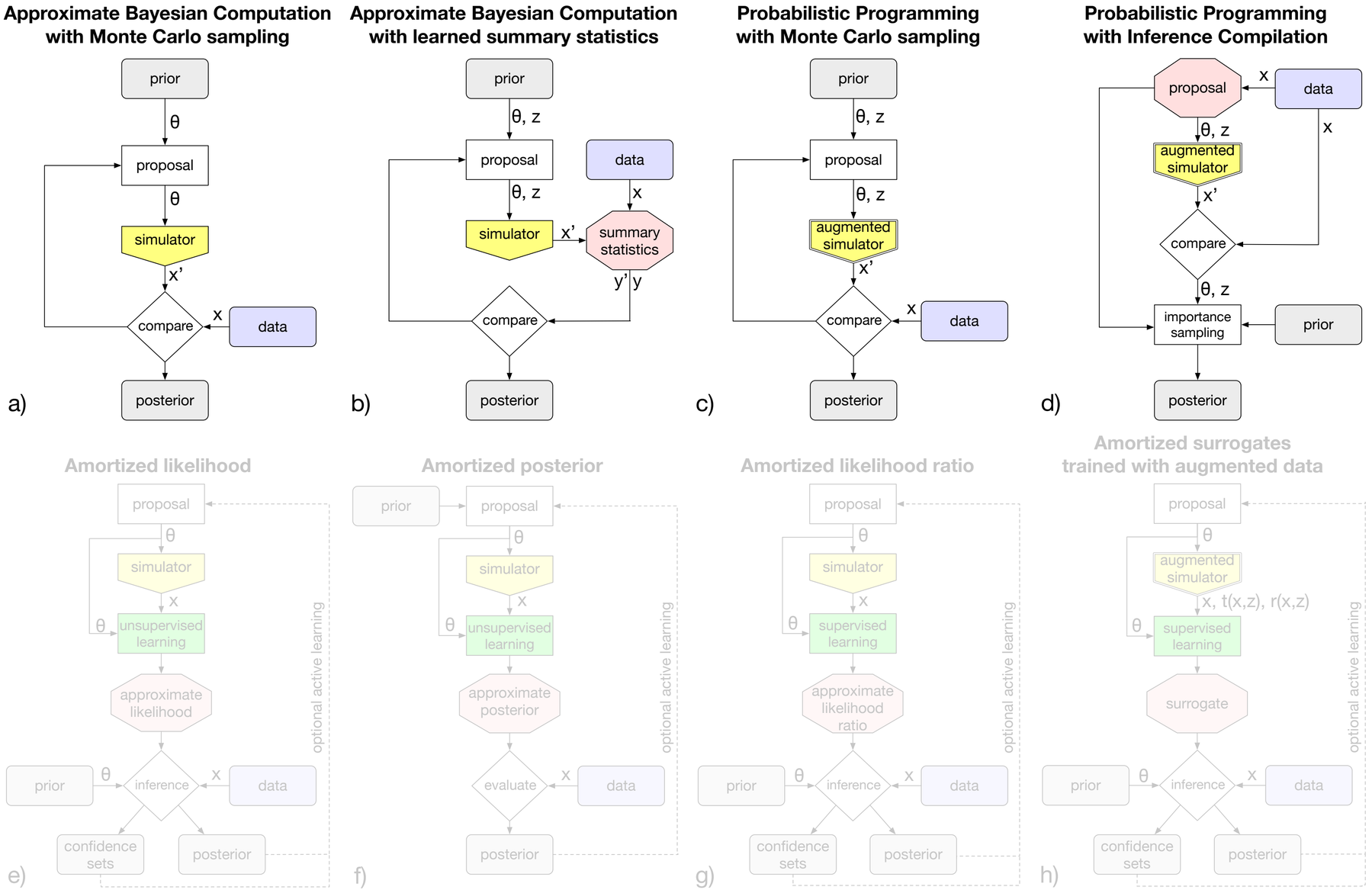


Fig. 3. Overview of different approaches to simulation-based inference.

From the review

[Cranmer, J. Brehmer, G. Louppe, PNAS (2020), arXiv:1911.01429]

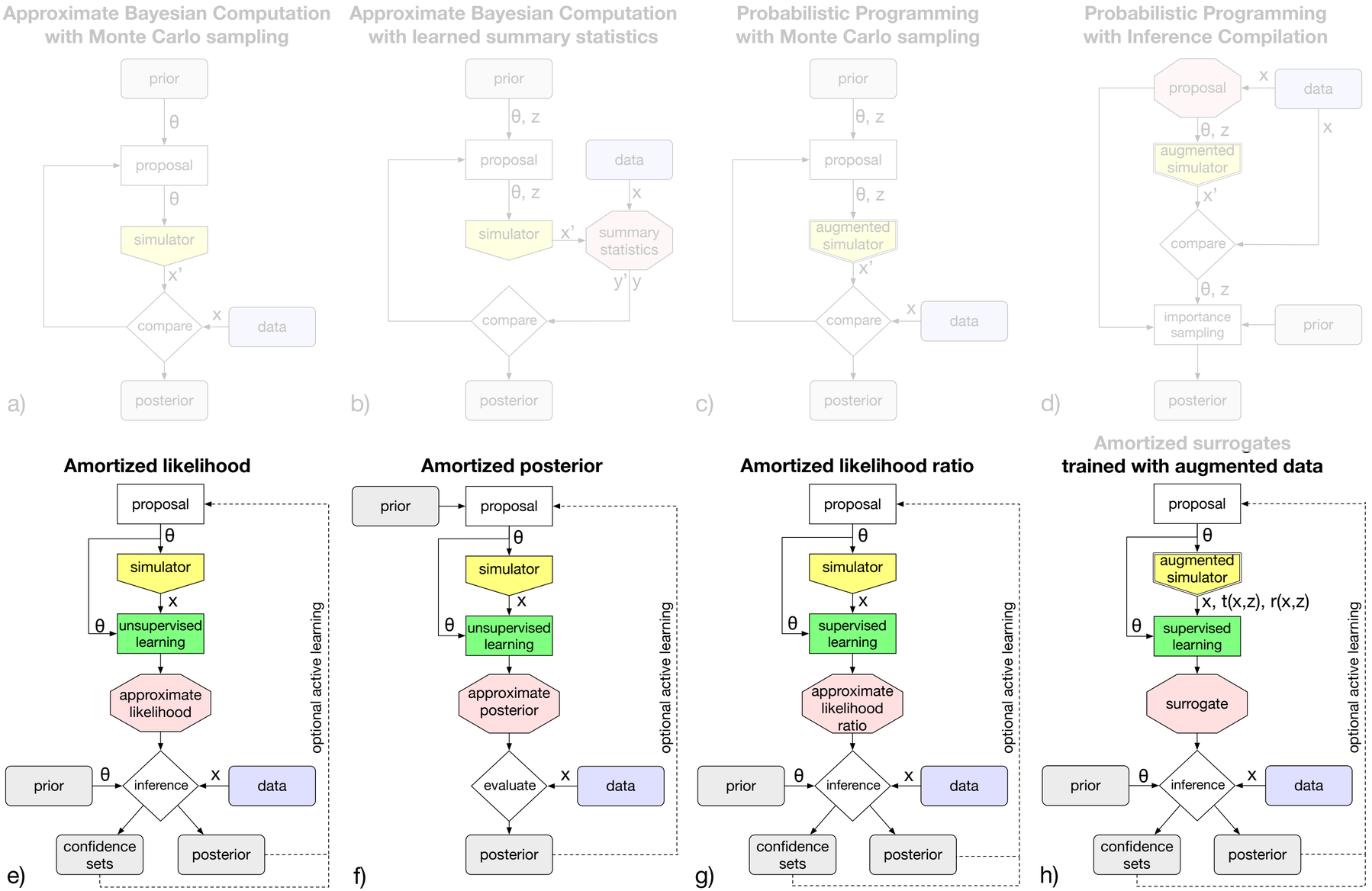


Fig. 3. Overview of different approaches to simulation-based inference.

From the review

[Cranmer, J. Brehmer, G. Louppe, PNAS (2020), arXiv:1911.01429]

Fig. 3. Overview of different approaches to simulation-based inference.

From the review

[Cranmer, J. Brehmer, G. Louppe, PNAS (2020), arXiv:1911.01429]

Fig. 3. Overview of different approaches to simulation-based inference.

From the review

[Cranmer, J. Brehmer, G. Louppe, PNAS (2020), arXiv:1911.01429]

Fig. 3. Overview of different approaches to simulation-based inference.

From the review

[Cranmer, J. Brehmer, G. Louppe, PNAS (2020), arXiv:1911.01429]

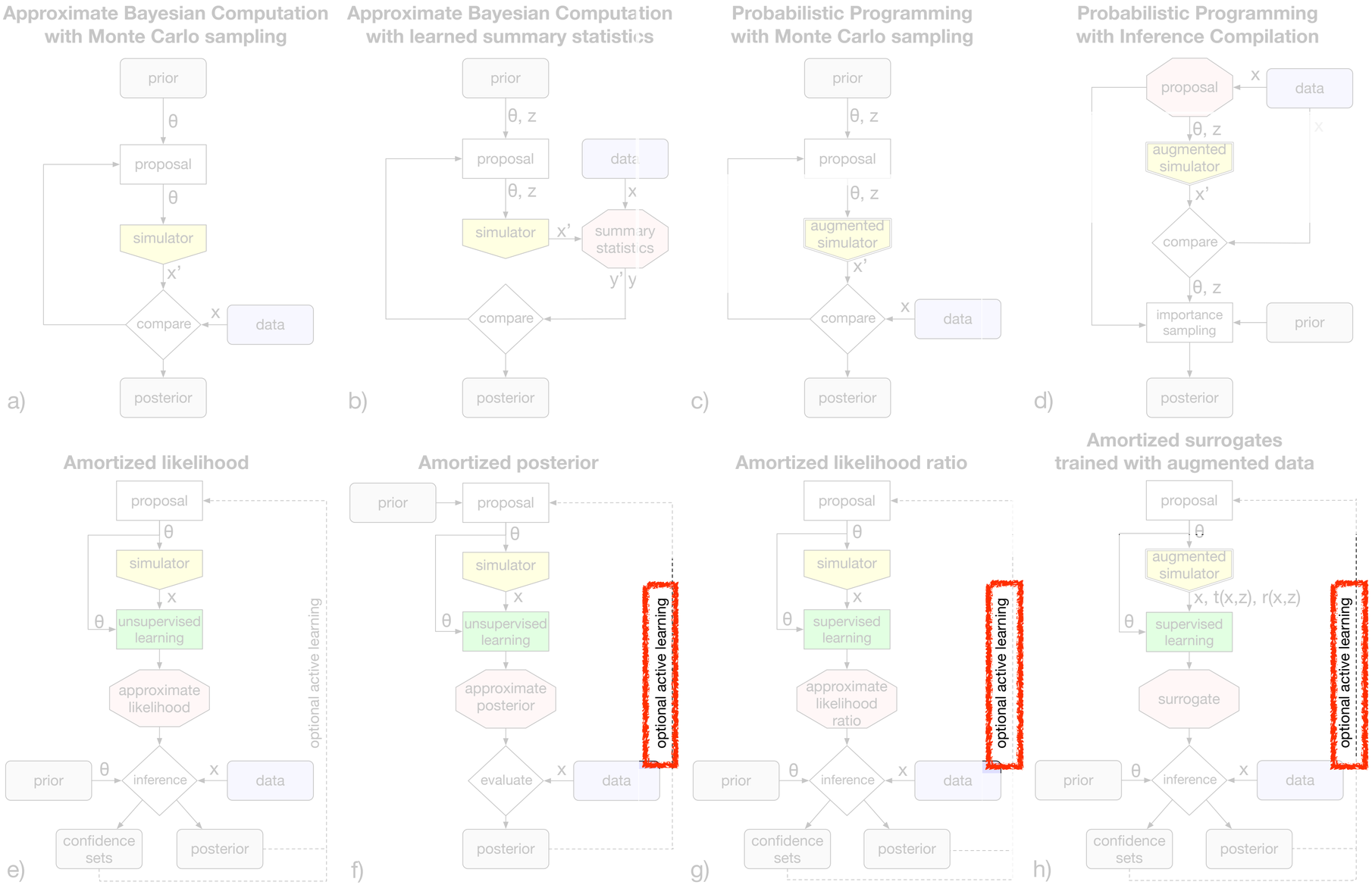


Fig. 3. Overview of different approaches to simulation-based inference.

Probabilistic Programming

[Cranmer, J. Brehmer, G. Louppe, PNAS (2020), arXiv:1911.01429]

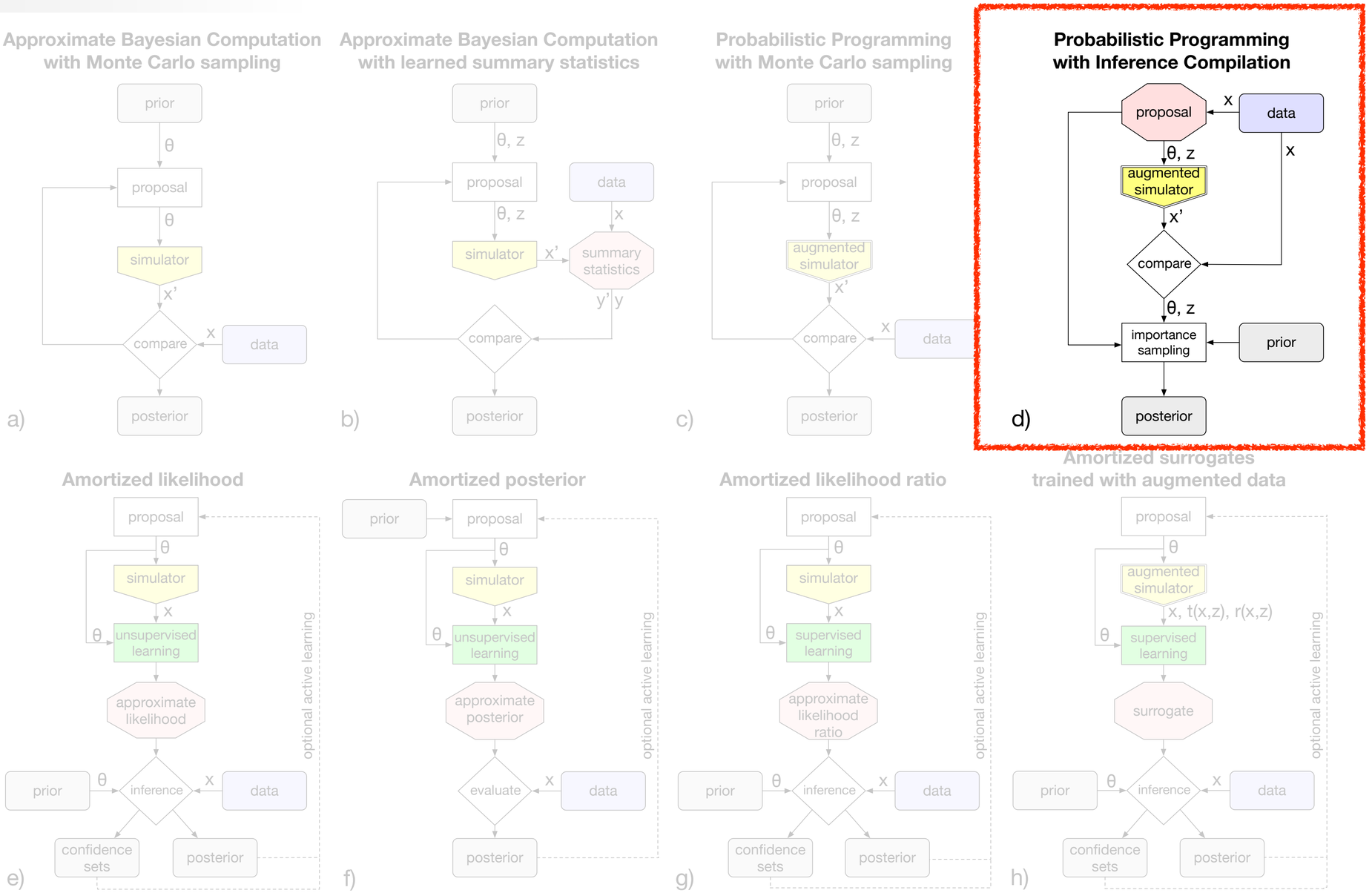
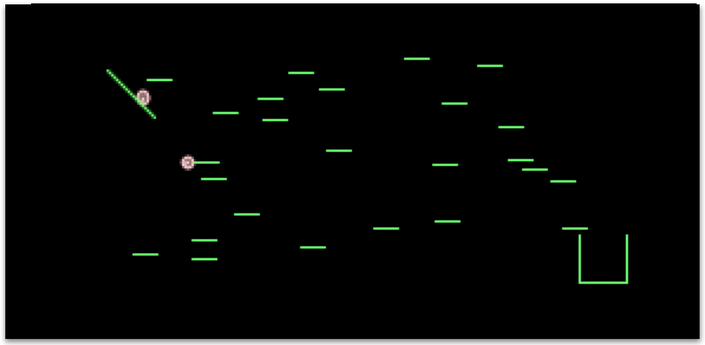
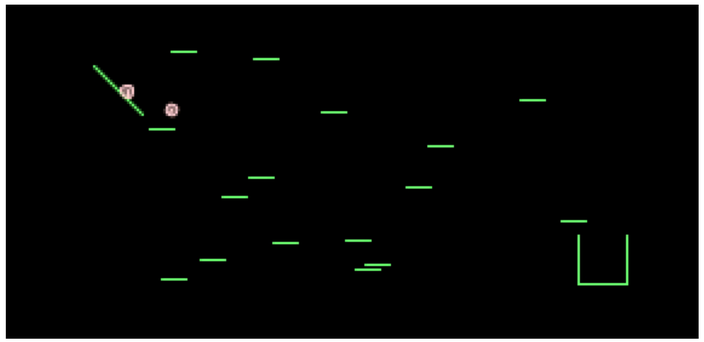


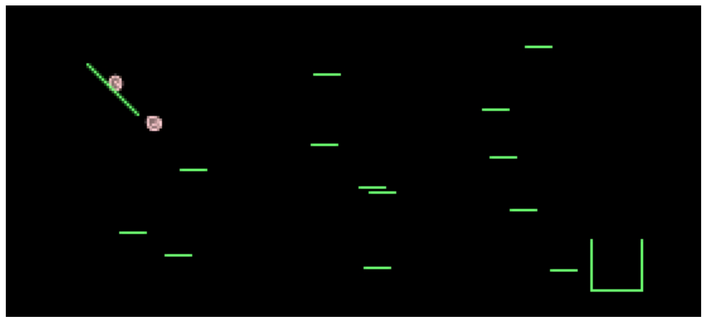
Fig. 3. Overview of different approaches to simulation-based inference.

Probabilistic Programming Example

```
(defquery arrange-bumpers []
    (let [number-of-bumpers (sample (poisson 20))
          bumpydist (uniform-continuous 0 10)
          bumpxdist (uniform-continuous -5 14)
          bumper-positions (repeatedly
                            number-of-bumpers
                            #(vector (sample bumpxdist)
                                     (sample bumpydist)))

          ;; code to simulate the world
          world (create-world bumper-positions)
          end-world (simulate-world world)
          balls (:balls end-world)

          ;; how many balls entered the box?
          num-balls-in-box (balls-in-box end-world)]

      {:balls balls
       :num-balls-in-box num-balls-in-box
       :bumper-positions bumper-positions}))
```





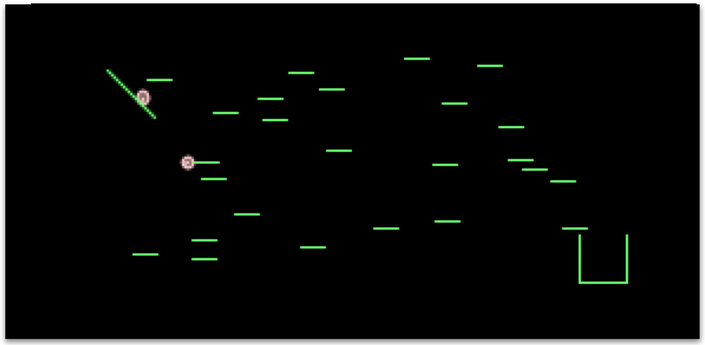


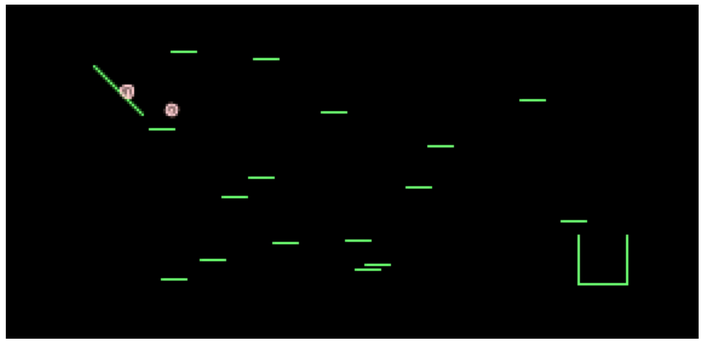
3 examples generated from simulator

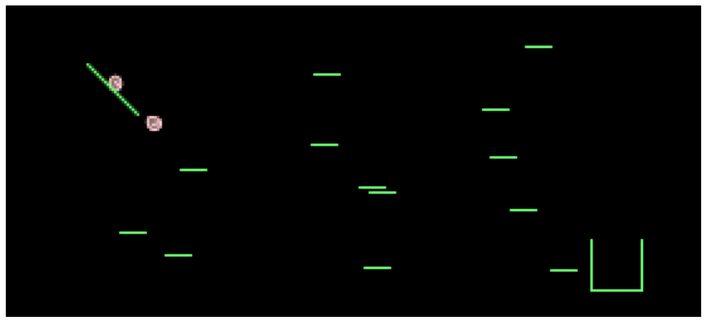
[slides, Frank Wood]

Probabilistic Programming Example

```
(defquery arrange-bumpers []
    (let [number-of-bumpers (sample (poisson 20))
          bumpydist (uniform-continuous 0 10)
          bumpxdist (uniform-continuous -5 14)
          bumper-positions (repeatedly
                            number-of-bumpers
                            #(vector (sample bumpxdist)
                                     (sample bumpydist)))

          ;; code to simulate the world
          world (create-world bumper-positions)
          end-world (simulate-world world)
          balls (:balls end-world)

          ;; how many balls entered the box?
          num-balls-in-box (balls-in-box end-world)]

      {:balls balls
       :num-balls-in-box num-balls-in-box
       :bumper-positions bumper-positions}))
```





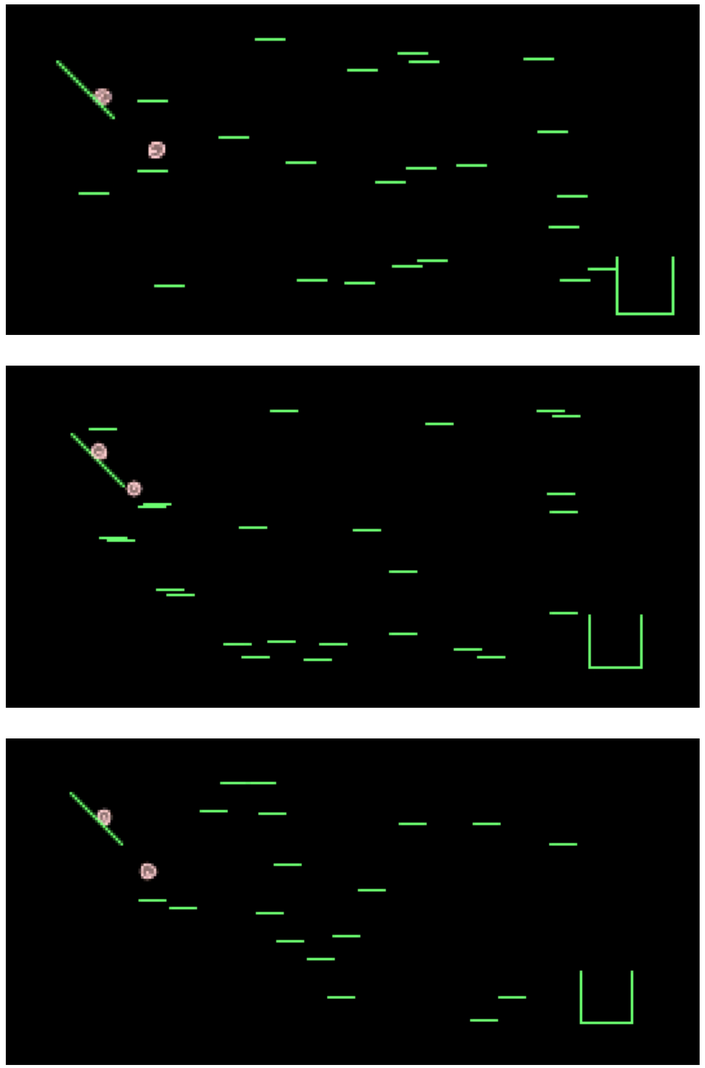


3 examples generated from simulator

[slides, Frank Wood]

Probabilistic Programming Example

```
(defquery arrange-bumpers []
    (let [number-of-bumpers (sample (poisson 20))
          bumpydist (uniform-continuous 0 10)
          bumpxdist (uniform-continuous -5 14)
          bumper-positions (repeatedly
                            number-of-bumpers
                            #(vector (sample bumpxdist)
                                     (sample bumpydist)))

          ;; code to simulate the world
          world (create-world bumper-positions)
          end-world (simulate-world world)
          balls (:balls end-world)

          ;; how many balls entered the box?
          num-balls-in-box (balls-in-box end-world)

          obs-dist (normal 4 0.1)]

      (observe obs-dist num-balls-in-box)
```

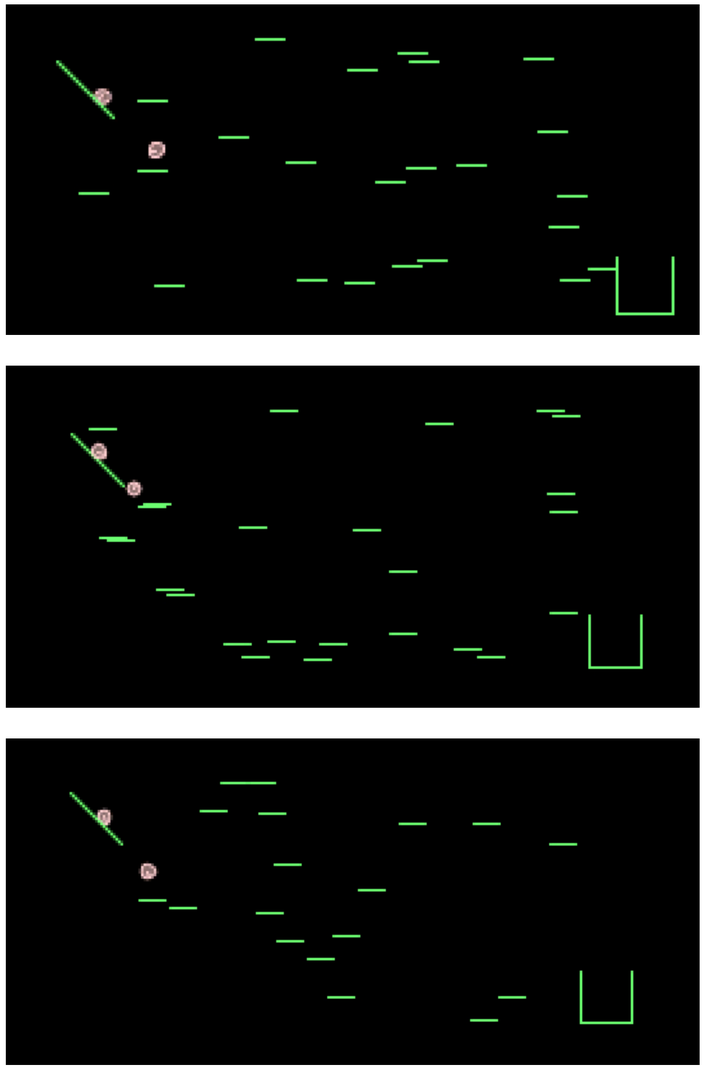


3 examples generated from simulator
conditioned on ~20% of balls land in box

[slides, Frank Wood]

Probabilistic Programming Example

```
(defquery arrange-bumpers []
    (let [number-of-bumpers (sample (poisson 20))
          bumpydist (uniform-continuous 0 10)
          bumpxdist (uniform-continuous -5 14)
          bumper-positions (repeatedly
                            number-of-bumpers
                            #(vector (sample bumpxdist)
                                     (sample bumpydist)))

          ;; code to simulate the world
          world (create-world bumper-positions)
          end-world (simulate-world world)
          balls (:balls end-world)

          ;; how many balls entered the box?
          num-balls-in-box (balls-in-box end-world)

          obs-dist (normal 4 0.1)]

      (observe obs-dist num-balls-in-box)
```



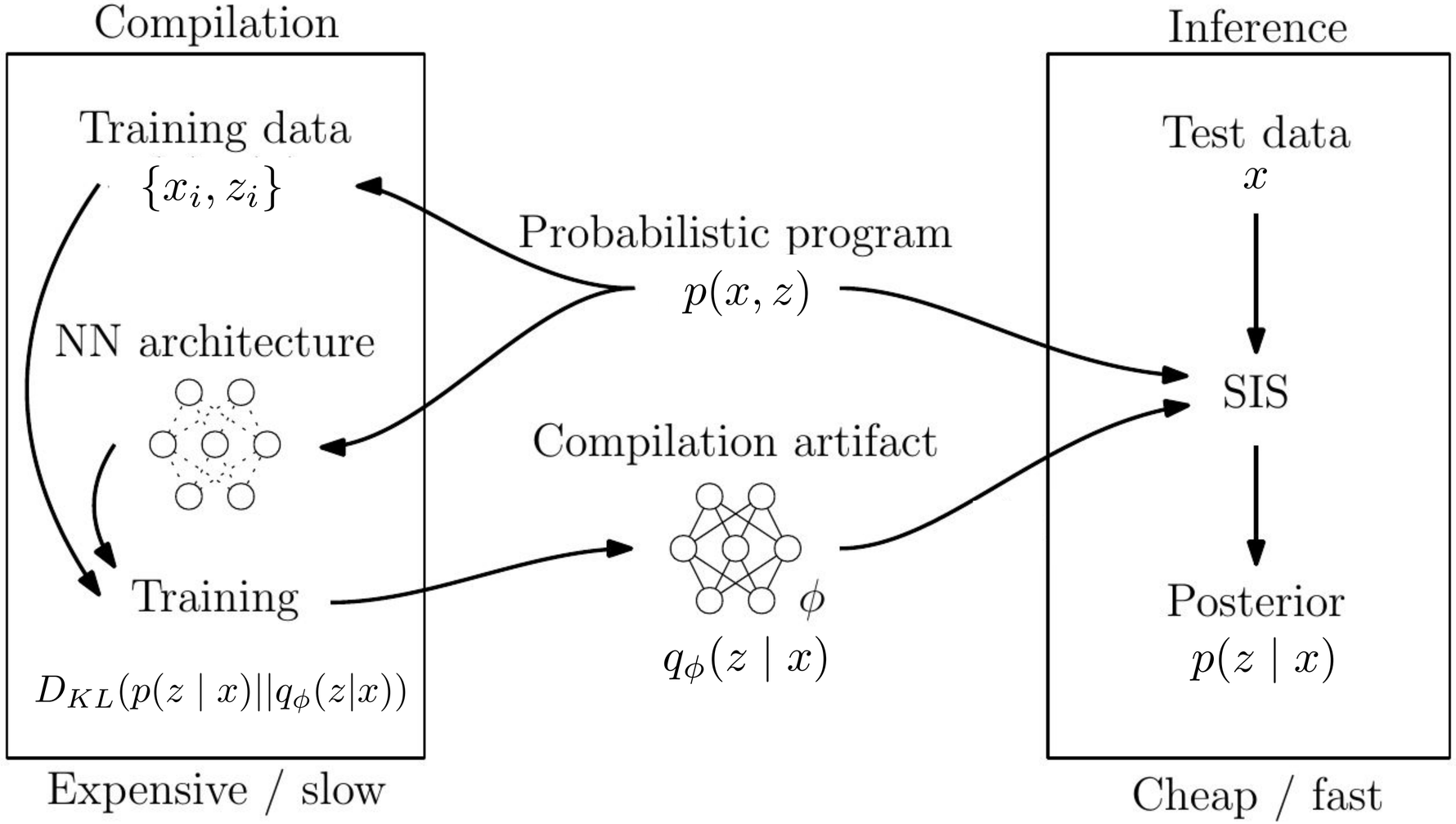
3 examples generated from simulator
conditioned on ~20% of balls land in box

[slides, Frank Wood]

Inference Compilation

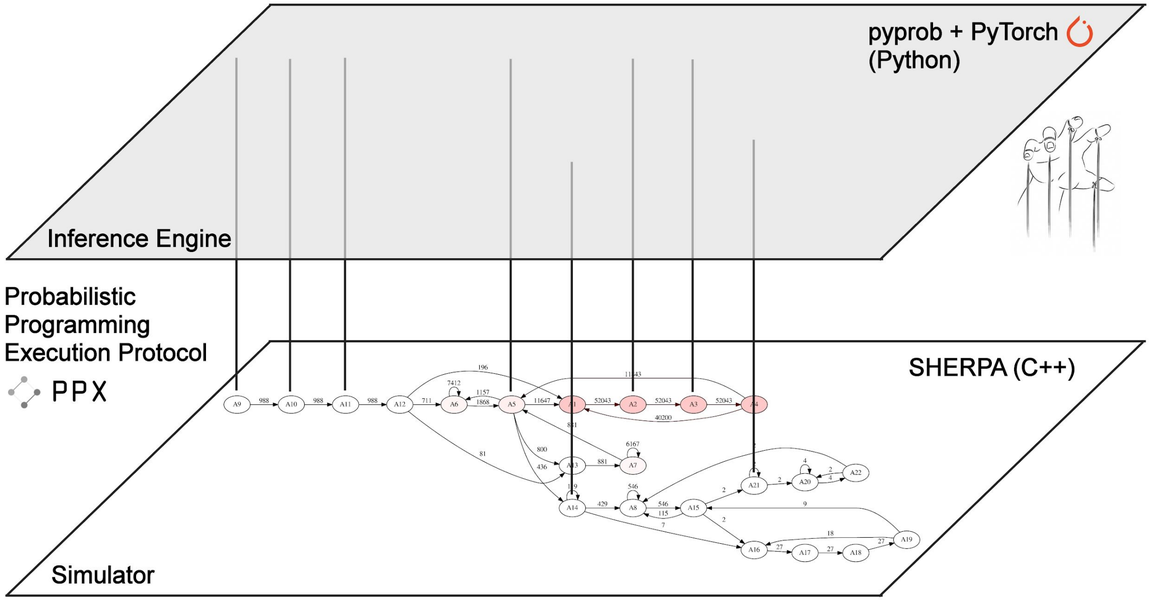
Le, Baydin, Wood AISTATS 2017, arXiv:1610.09900

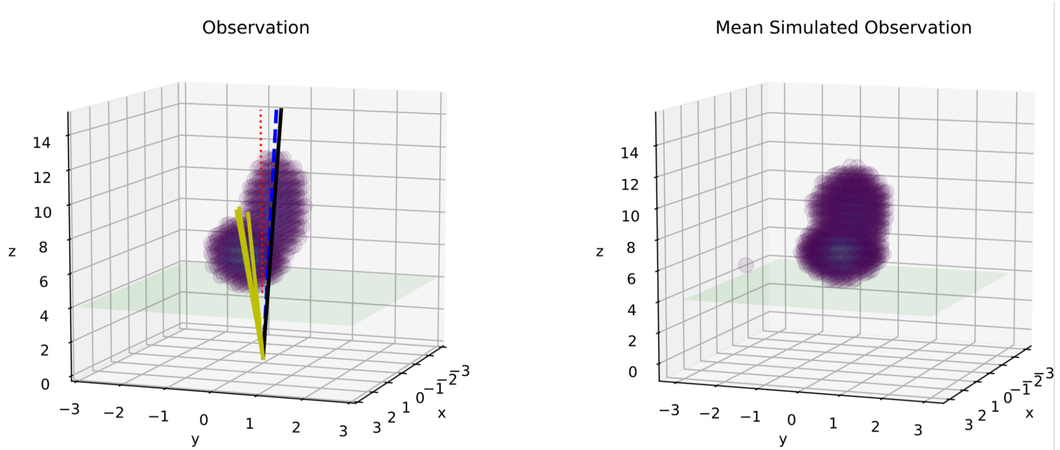
Hijack the random number generators and use NN's to learn $q_\phi(z \mid x)$ and then perform a *very* smart type of importance sampling over structured latent space of stack traces.



G. Baydin, et al SC19 arXiv:1907.03382
G. Baydin, et al. NeurIPS 2019 arXiv:1807.07706

Previously had to use a special purpose probabilistic programming language. With **ppx** protocol, we decouple inference engine & control existing simulator.



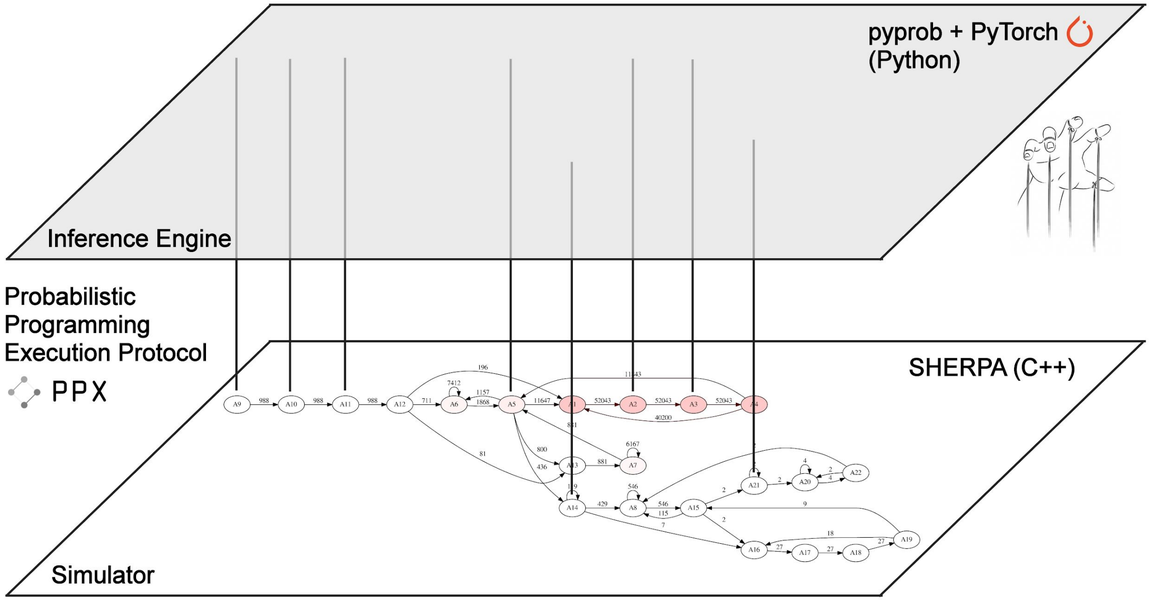


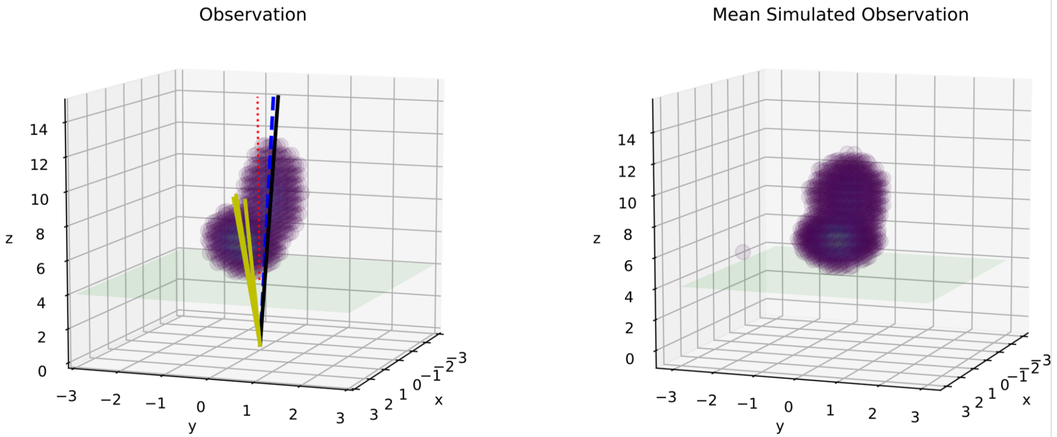
- **Augment** real-world physics simulator
  (C++, 1M lines of code)
- 3DCNN-LSTM architecture for $q_\phi(z \mid x)$
  (Stack traces with Dim[z] ranging from 100 — 2,000)
- Inference is embarrassingly parallelizable unlike MCMC. 230x speedup

Atılım Güneş Baydin
Bradley Gram-Hansen
Lukas Heinrich
Kyle Cranmer
Frank Wood
Andreas Munk
Saeid Naderiparizi
Wahid Bhimji
Jialin Liu
Prabhat
Gilles Louppe
Lei Shao
Larry Meadows

G. Baydin, et al SC19 arXiv:1907.03382
G. Baydin, et al. NeurIPS 2019 arXiv:1807.07706

Previously had to use a special purpose probabilistic programming language. With **ppx** protocol, we decouple inference engine & control existing simulator.

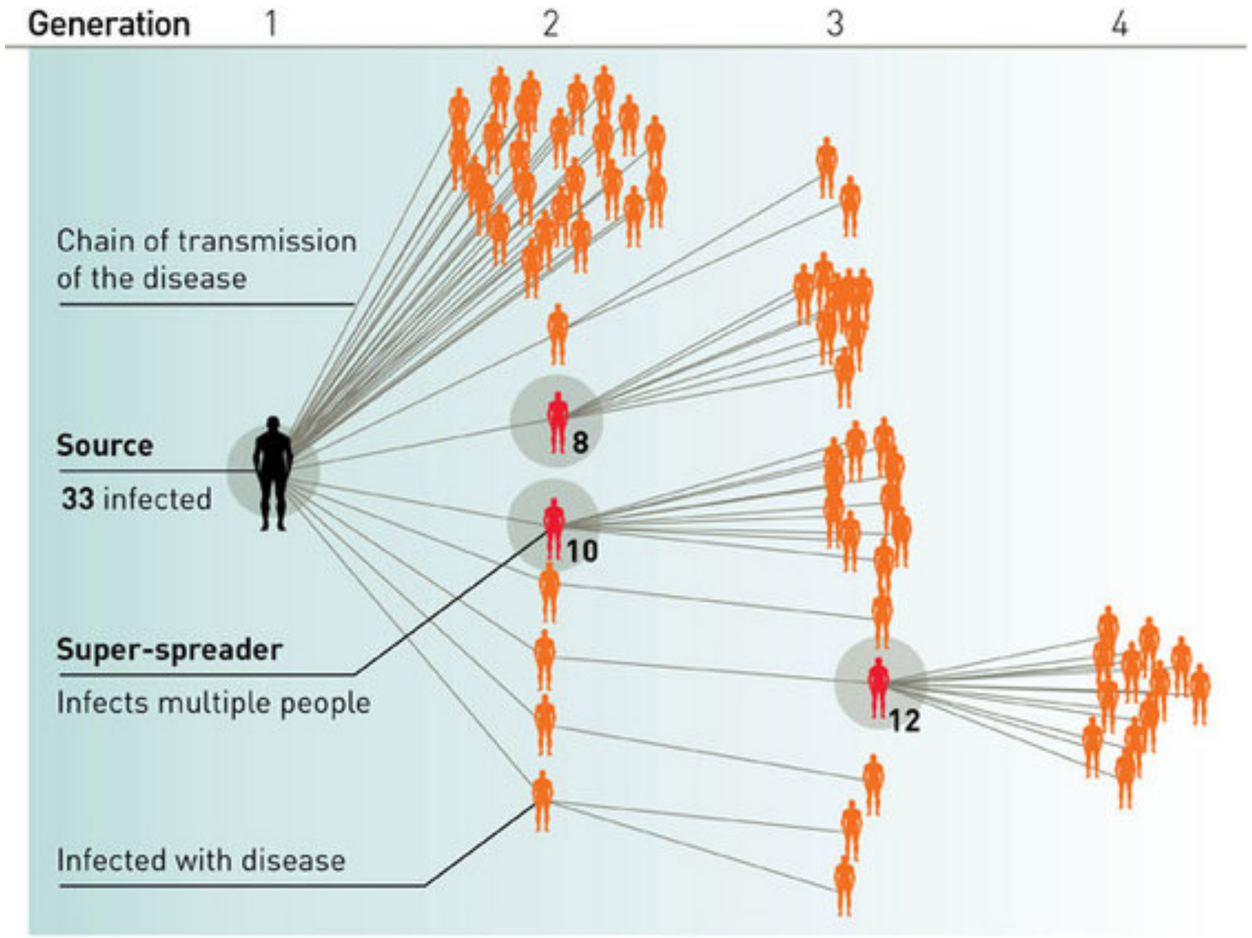




- **Augment** real-world physics simulator
  (C++, 1M lines of code)
- 3DCNN-LSTM architecture for $q_\phi(z \mid x)$
  (Stack traces with Dim[z] ranging from 100 — 2,000)
- Inference is embarrassingly parallelizable unlike MCMC. 230x speedup

Atılım Güneş Baydin
Bradley Gram-Hansen
Lukas Heinrich
Kyle Cranmer
Frank Wood
Andreas Munk
Saeid Naderiparizi
Wahid Bhimji
Jialin Liu
Prabhat
Gilles Louppe
Lei Shao
Larry Meadows

Epidemiology & population Genetics



Simulation-Based Inference for Global Health Decisions

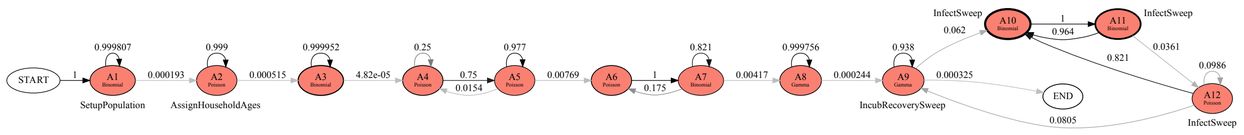


Figure 1: Latent probabilistic structure uncovered using `PyProb` from the Imperial College `CovidSim` simulator run on Malta, demonstrating the first step in working with this simulator as a probabilistic program. Uniform distributions are omitted for simplicity.

Simulation-Based Inference for Global Health Decisions

Christian Schroeder de Witt [1] Bradley Gram-Hansen [1] Nantas Nardelli [1]
Andrew Gambardella [1] Rob Zinkov [1] Puneet Dokania [1] N. Siddharth [1]
Ana Belen Espinosa-Gonzalez [2] Ara Darzi [2] Philip Torr [1] Atılım Güneş Baydin [1]

https://arxiv.org/abs/2005.07062

PLANNING AS INFERENCE IN EPIDEMIOLOGICAL DYNAMICS MODELS

A PREPRINT

Frank Wood[1,3,4], Andrew Warrington[2], Saeid Naderiparizi[1], Christian Weilbach[1], Vaden Masrani[1],
William Harvey[1], Adam Ścibior[1], Boyan Beronov[1], and Ali Nasseri[1]

[1]Department of Computer Science, University of British Columbia
[2]Department of Engineering Science, University of Oxford
[3]MILA
[4]CIFAR AI Chair
{fwood,awarring,saeidnp,weilbach,vadmas,wsgh,ascibior,beronov}@cs.ubc.ca, ali.nasseri@ubc.ca

https://arxiv.org/abs/2003.13221

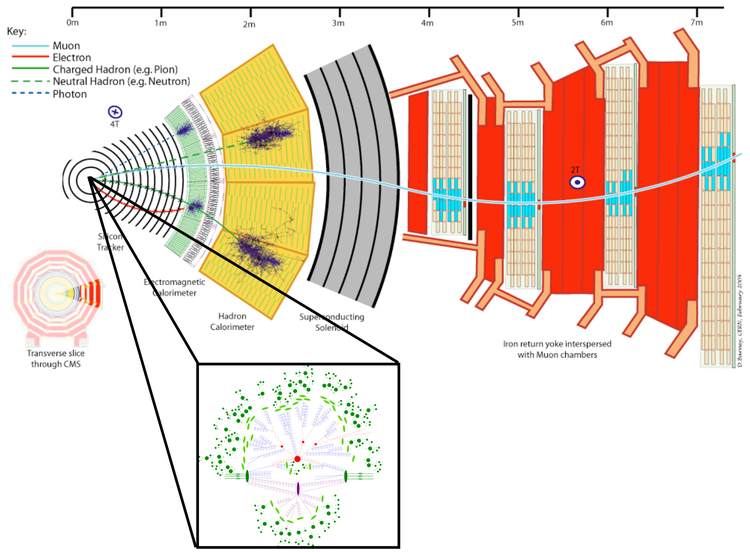
Hijacking Malaria Simulators with Probabilistic Programming

Bradley J. Gram-Hansen [*1] Christian Schröder de Witt [*1]
Tom Rainforth [2] Philip H.S. Torr [1] Yee Whye Teh [2] Atılım Güneş Baydin [1]

https://arxiv.org/abs/1905.12432

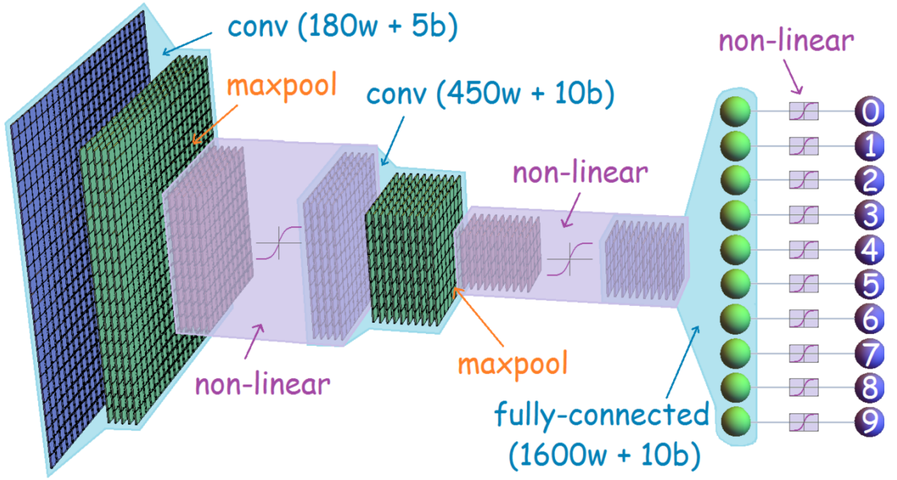
Two approaches simulation-based inference

Use simulator
(much more efficiently)



- Approximate Bayesian Computation (ABC)
- Probabilistic Programming
- Adversarial Variational Optimization

Learn simulator
(with deep learning)



- Likelihood ratio trick (with classifiers)
- Conditional density estimate (with normalizing flows)
- Learned summary statistics

[image credit: A.P. Goucher]

Different targets

Learn a likelihood ratio or density ratio with a classifier

- Neural Ratio Estimation [NRE]
- likelihood ratio to arbitrary reference $r(x;\theta) = \frac{p(x \mid \theta)}{p_{\text{ref}}(x)}$ or between $r(x;\theta_0,\theta_1) = \frac{p(x \mid \theta_0)}{p(x \mid \theta_1)}$
- likelihood / evidence = posterior / prior $r(x;\theta) = \frac{p(x \mid \theta)}{p(x)} = \frac{p(\theta \mid x)}{p(\theta)}$

Learn the likelihood $p(x \mid \theta)$ with a conditional density estimate

- Neural Likelihood Estimation [NLE]

Learn the posterior $p(\theta \mid x)$ with a conditional density estimate

- Neural Posterior Estimation [NPE]

From the review

[Cranmer, J. Brehmer, G. Louppe, PNAS (2020), arXiv:1911.01429]

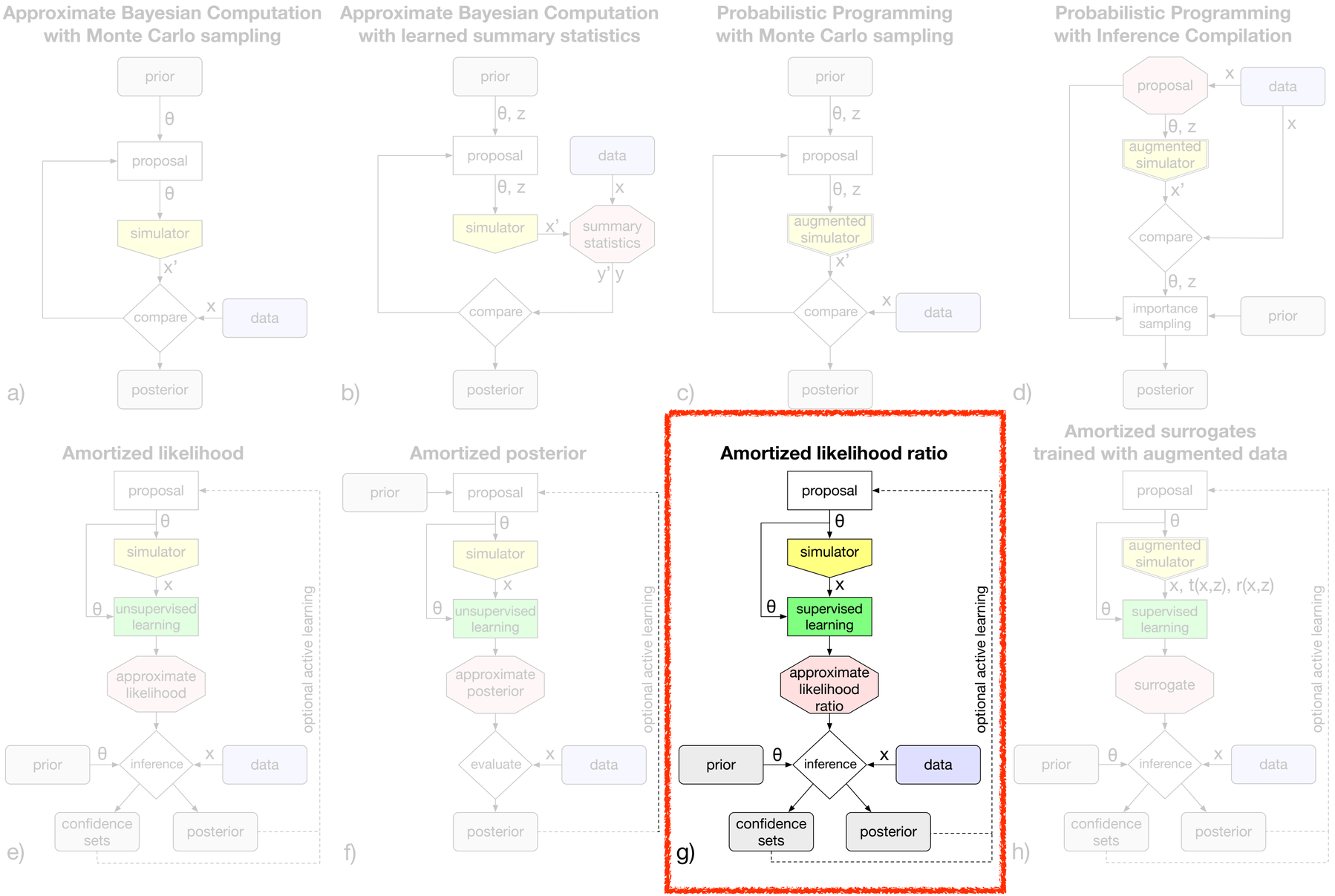


Fig. 3. Overview of different approaches to simulation-based inference.

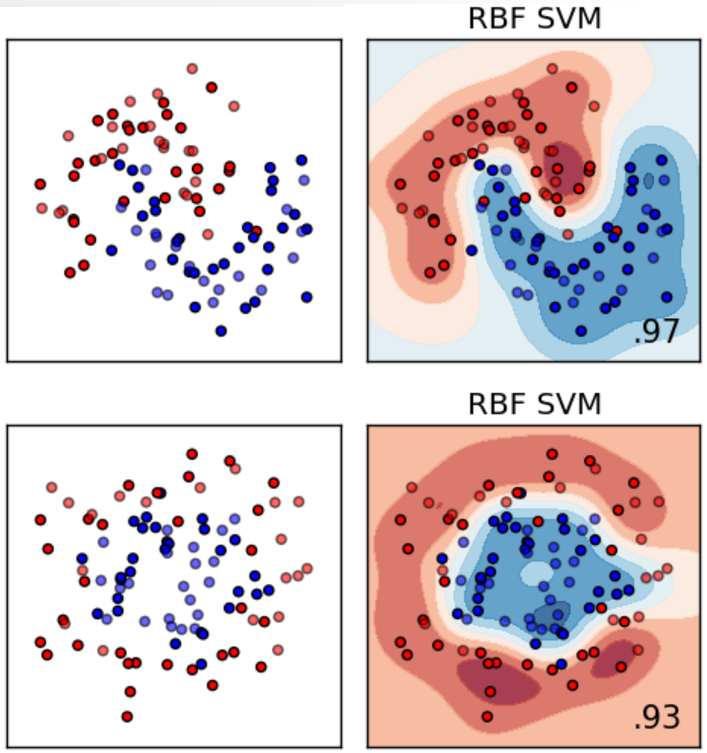
From the review

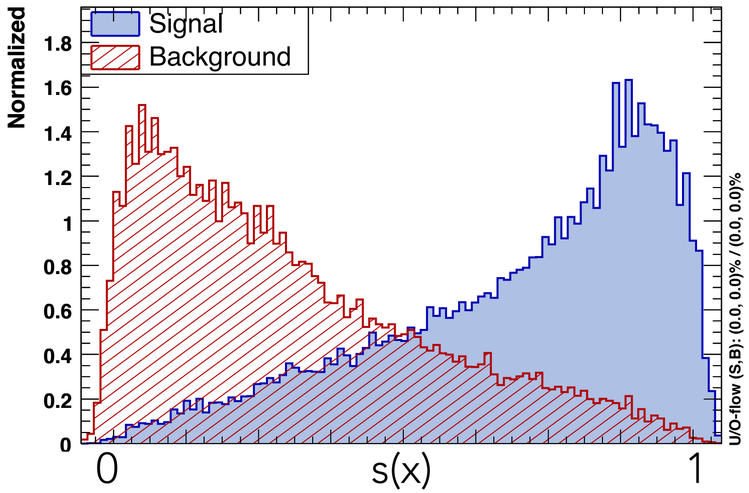
[Cranmer, J. Brehmer, G. Louppe, PNAS (2020), arXiv:1911.01429]

Fig. 3. Overview of different approaches to simulation-based inference.

Likelihood Ratio Trick

[Hastie et al., 2001; Sugiyama et al., 2012; Cranmer et. al., 2015]





- **binary classifier**: find function $s(x)$ that minimizes ***loss***:

$$L[s] = \mathbb{E}_{p(x|H_1)}[-\log s(x)] + \mathbb{E}_{p(x|H_0)}[-\log(1 - s(x))]$$

- i.e. approximate the optimal classifier

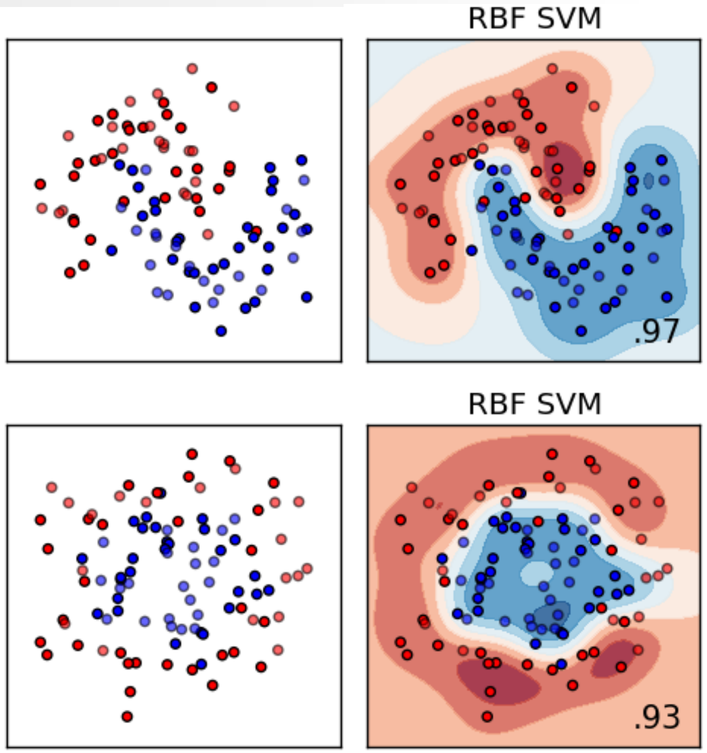
$$s(x) = \frac{p(x|H_1)}{p(x|H_0) + p(x|H_1)}$$

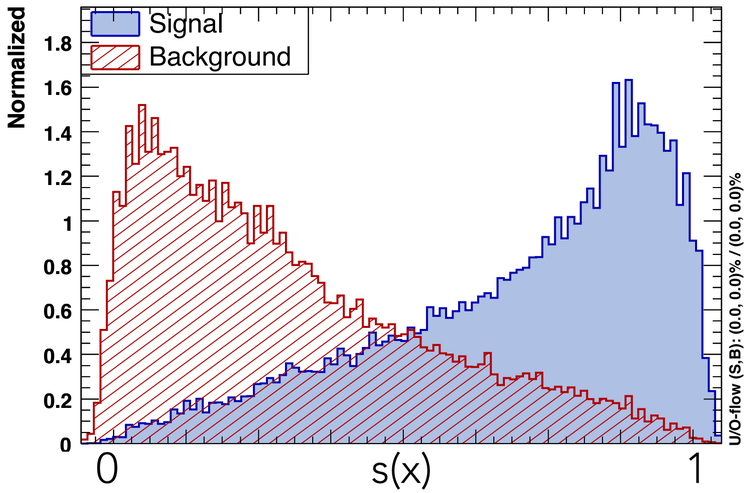
- which is 1-to-1 with the likelihood ratio

$$r(x) = \frac{p(x|H_1)}{p(x|H_0)} = 1 - \frac{1}{s(x)}$$

Likelihood Ratio Trick

[Hastie et al., 2001; Sugiyama et al., 2012; Cranmer et. al., 2015]





- **binary classifier**: find function $s(x)$ that minimizes ***loss***:

$$L[s] = \mathbb{E}_{p(x|H_1)}[-\log s(x)] + \mathbb{E}_{p(x|H_0)}[-\log(1-s(x))]$$

$$\approx \frac{1}{N}\sum_{i=1}^{N} -y_i \log s(x_i) - (1-y_i)\log(1-s(x_i))$$

- i.e. approximate the optimal classifier

$$s(x) = \frac{p(x|H_1)}{p(x|H_0) + p(x|H_1)}$$

- which is 1-to-1 with the likelihood ratio

$$r(x) = \frac{p(x|H_1)}{p(x|H_0)} = 1 - \frac{1}{s(x)}$$

Parametrizing the Likelihood Ratio Trick

Can do the same thing for any two points θ_0 & θ_1 in parameter space Θ.

$$r(x;\theta_0,\theta_1) = \frac{p(x \mid \theta_0)}{p(x \mid \theta_1)} = 1 - \frac{1}{s(x;\theta_0,\theta_1)}$$

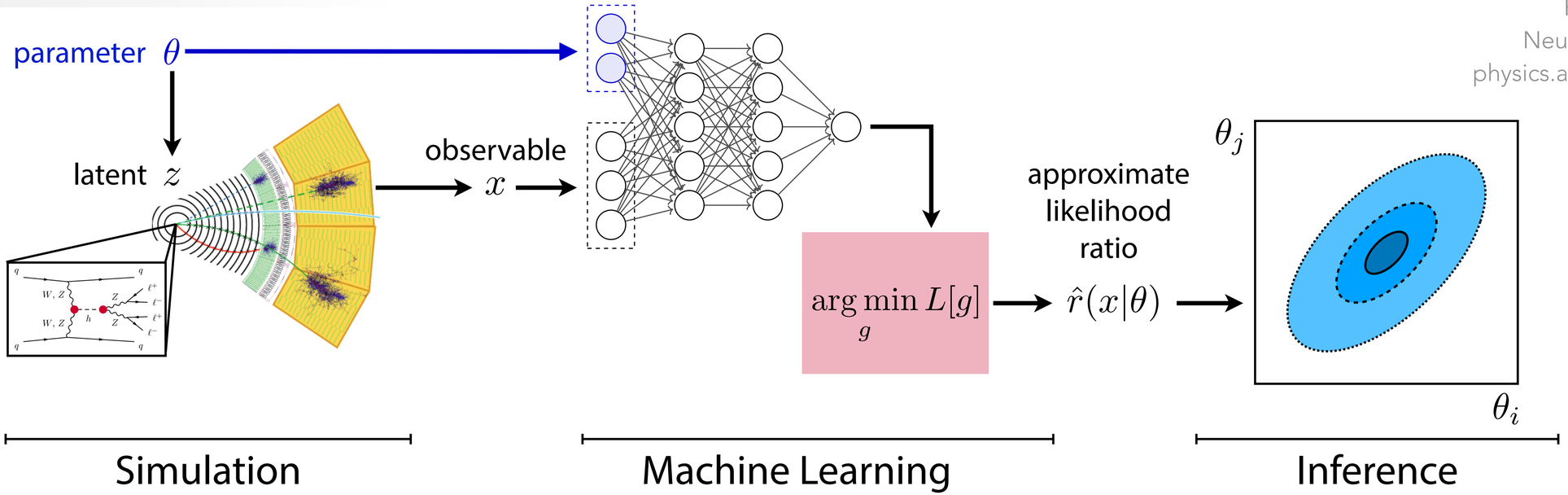
Or train to classify data from $p(x|\theta)$ versus some fixed reference $p_{\text{ref}}(x)$

$$r(x;\theta) = \frac{p(x|\theta)}{p_{\text{ref}}(x)} = 1 - \frac{1}{s(x;\theta)}$$

I call this a **parametrized classifier.**

Learning the likelihood ratio

Cranmer, Louppe, Pavez, arXiv:1506.02169
PNAS, arXiv:1805.12244
PRL, arXiv:1805.00013
PRD, arXiv:1805.00020
NeurIPS, arXiv:1808.00973
physics.aps.org/articles/v11/90



The **surrogate for the likelihood ratio** used for inference

A 2-stage process:

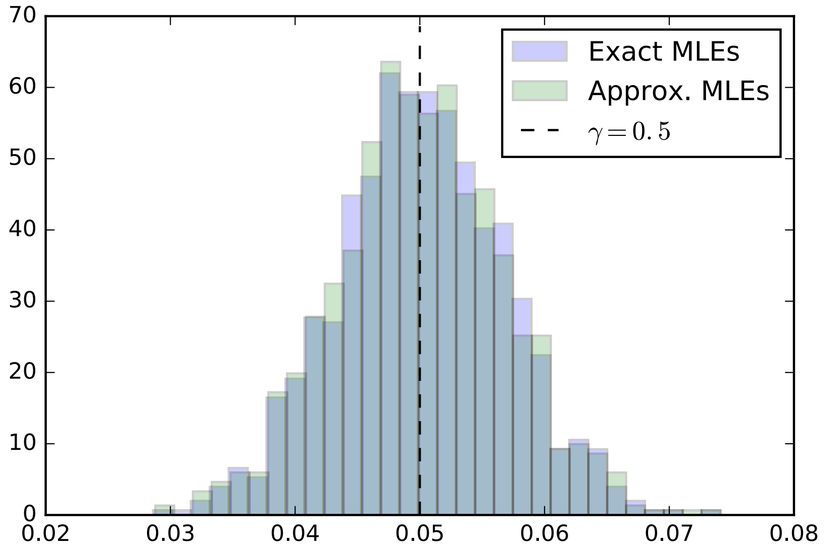
1. learning surrogate (**amortized**)
2. Inference on parameters of simulator (frequentist or Bayesian)

No Bayesian prior used for training, but one can use prior for inference.

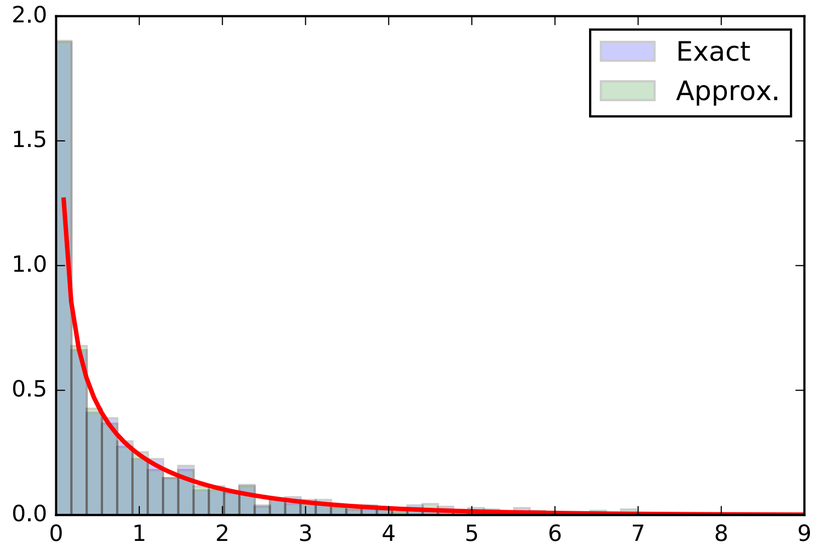
Amortized likelihood ratio

Once we've learned the likelihood ratio $r(x;\theta)$, we can apply it to any data x.

- unlike ABC, we pay biggest computational costs up front
- Great for calibrated frequentist confidence intervals with guaranteed coverage
- Here we repeat inference thousands of times & check asymptotic statistical theory



(a) Exact vs. approximated MLEs.

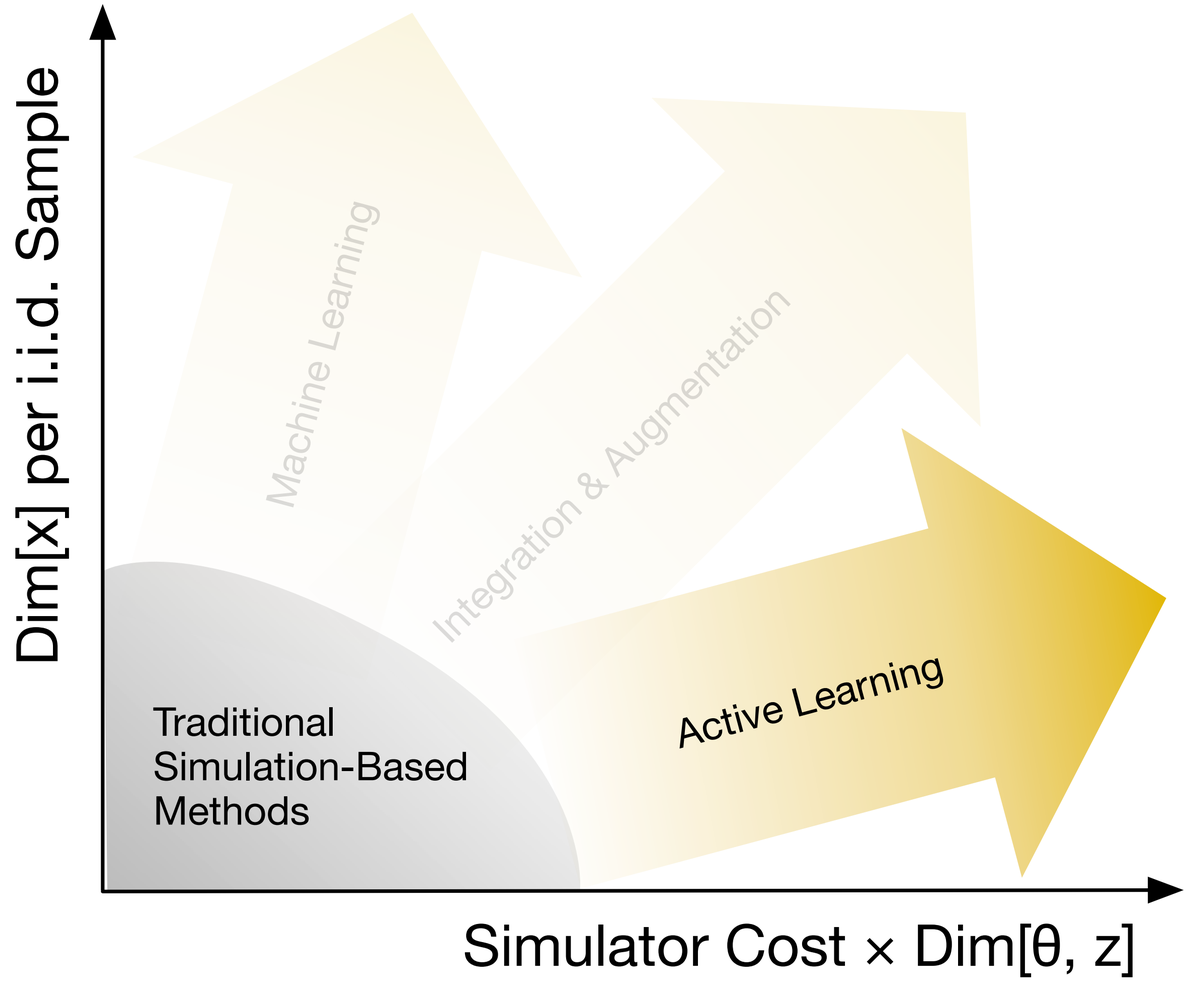


(b) $p(-2\log\Lambda(\gamma = 0.05)\,|\,\gamma = 0.05)$

Active learning and sequential methods

Can we learn more efficiently for a fixed simulation budget ?

- What if we are smart about where we run the simulator?



From the review

[Cranmer, J. Brehmer, G. Louppe, PNAS (2020), arXiv:1911.01429]

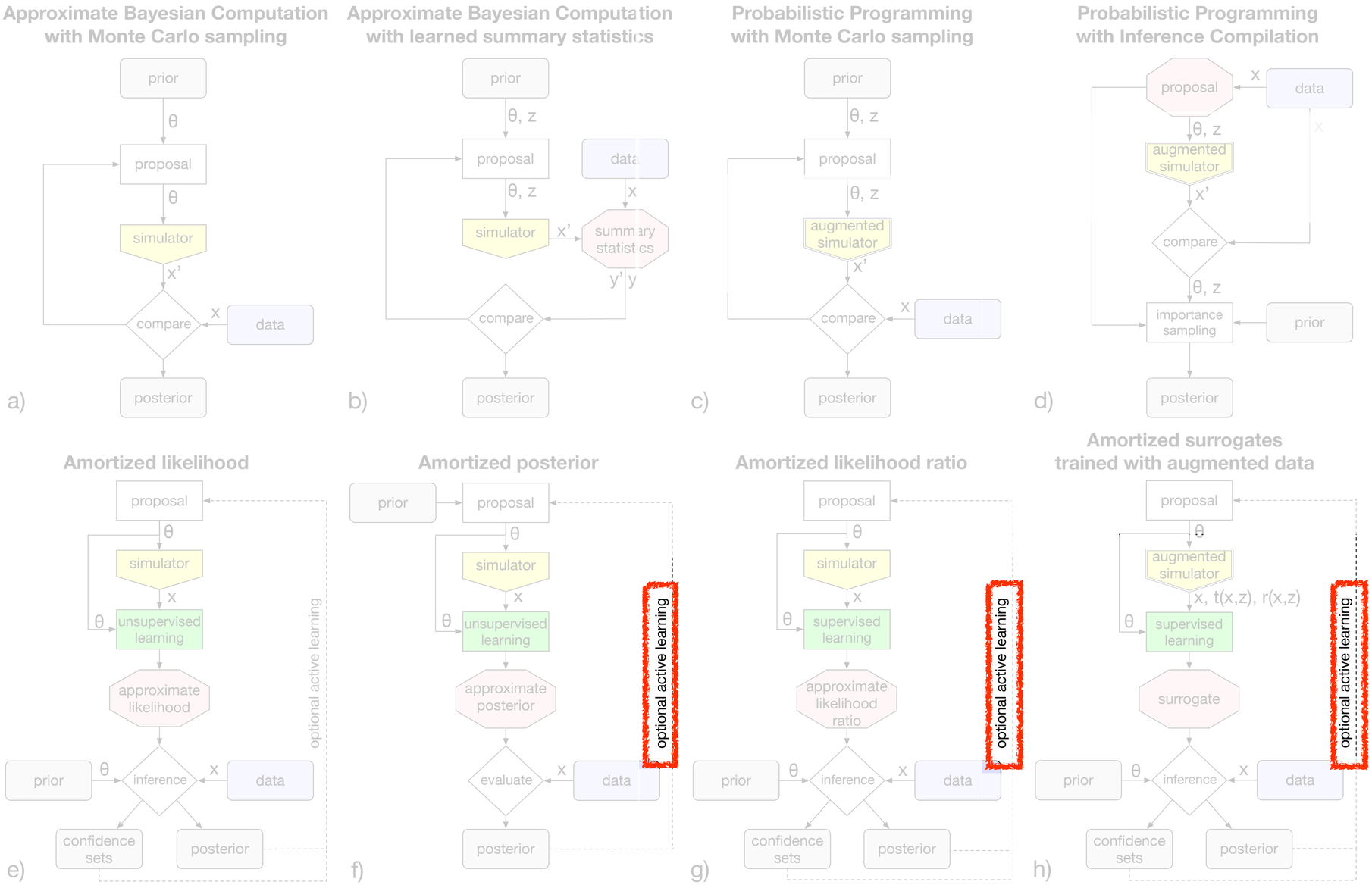


Fig. 3. Overview of different approaches to simulation-based inference.

Sequential Methods

When the posterior concentrates significantly compared to the prior, then we don't really need to estimate the likelihood accurately everywhere

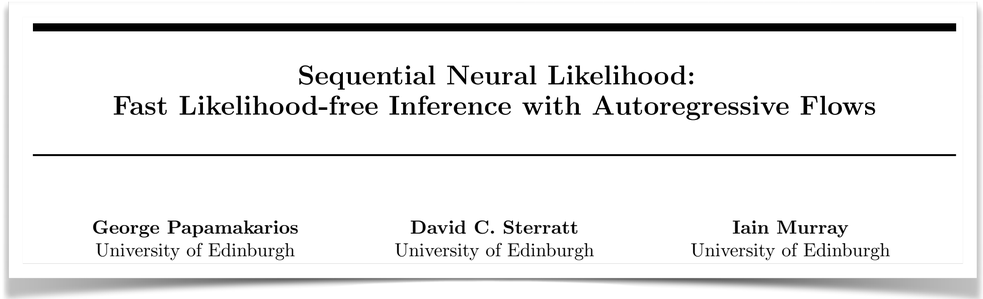
- Instead, want to estimate likelihood or posterior only in the **relevant regions** of parameter / data space
- Motivates active learning / sequential techniques
- **Iteratively** estimate posterior $\tilde{p}(\theta \mid x_0)$, sample $\theta_n \sim \tilde{p}(\theta \mid x_0)$, $x_n \sim p(x \mid \theta_n)$, and then **refine**

Sequential Neural Likelihood Estimation [SNLE]

Sequential Neural Posterior Estimation [SNPE]

Sequential Neural Ratio Estimation [SNRE]

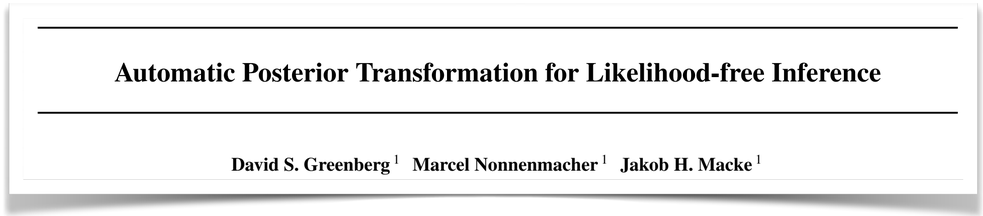
- Various sequential strategies


Sequential Neural Likelihood:
Fast Likelihood-free Inference with Autoregressive Flows

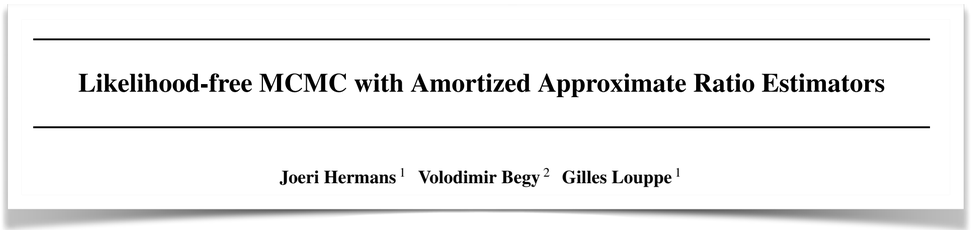
George Papamakarios
University of Edinburgh

David C. Sterratt
University of Edinburgh

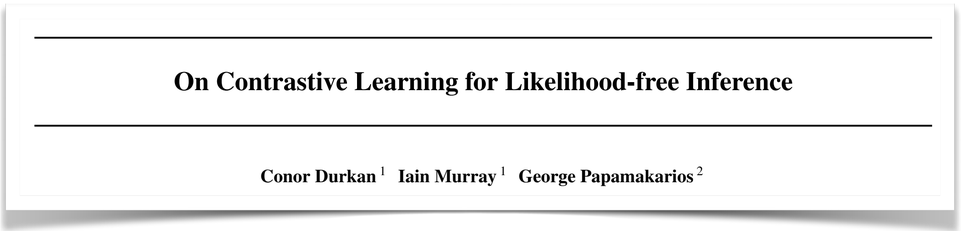
Iain Murray
University of Edinburgh


Automatic Posterior Transformation for Likelihood-free Inference

David S. Greenberg [1] Marcel Nonnenmacher [1] Jakob H. Macke [1]


Likelihood-free MCMC with Amortized Approximate Ratio Estimators

Joeri Hermans [1] Volodimir Begy [2] Gilles Louppe [1]


On Contrastive Learning for Likelihood-free Inference

Conor Durkan [1] Iain Murray [1] George Papamakarios [2]

$$\tilde{p}(\theta|x) = p(\theta|x)\frac{\tilde{p}(\theta)\,p(x)}{p(\theta)\,\tilde{p}(x)}$$

Sequential Neural Posterior Estimation (SNPE, Papamakarios & Murray, 2016; Lueckmann et al., 2017; Greenberg et al., 2019),

Sequential Methods

When the posterior concentrates significantly compared to the prior, then we don't really need to estimate the likelihood accurately everywhere

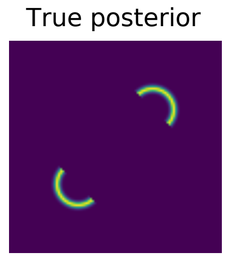
- Instead, want to estimate likelihood or posterior only in the **relevant regions** of parameter / data space
- Motivates active learning / sequential techniques
- **Iteratively** estimate posterior $\tilde{p}(\theta \mid x_0)$, sample $\theta_n \sim \tilde{p}(\theta \mid x_0)$, $x_n \sim p(x \mid \theta_n)$, and then **refine**

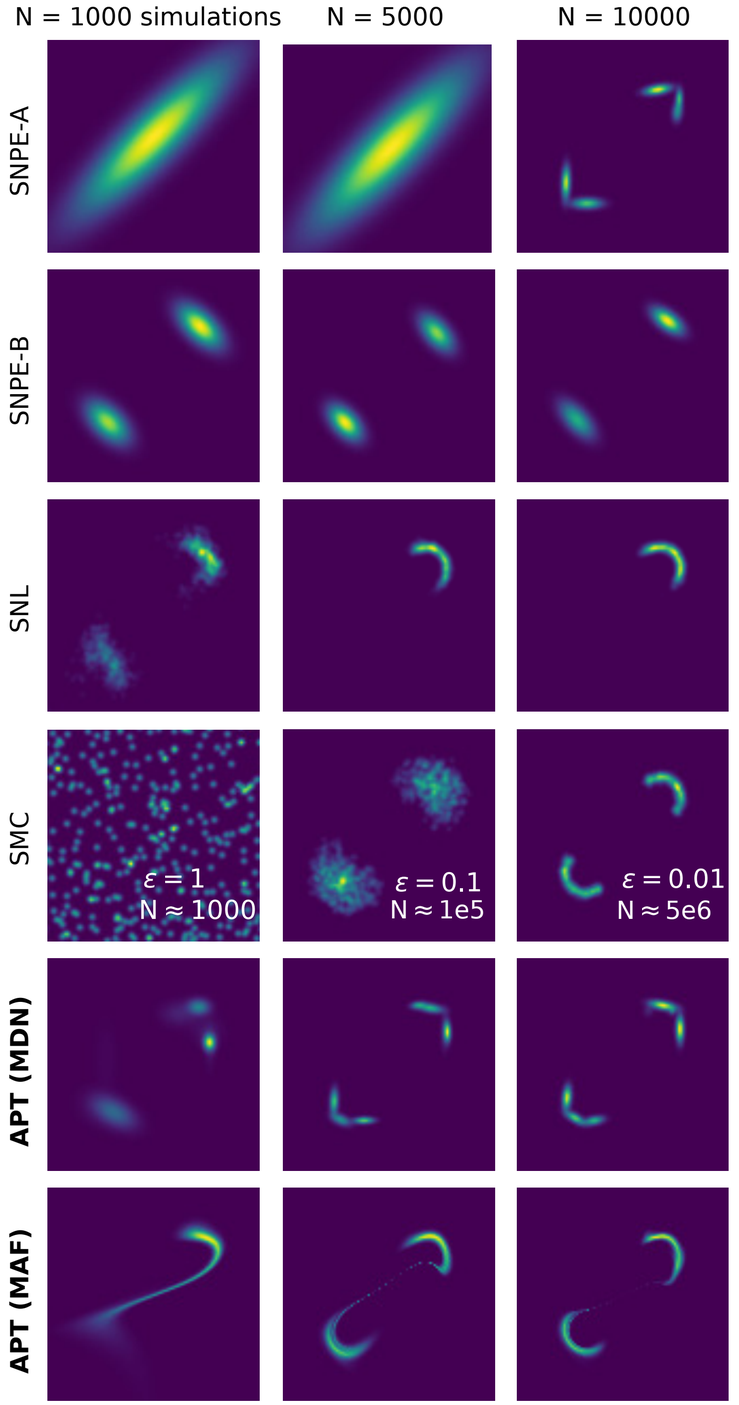
Sequential Neural Likelihood Estimation [SNLE]

Sequential Neural Posterior Estimation [SNPE]

Sequential Neural Ratio Estimation [SNRE]

- Various sequential strategies





sbi: A toolkit for simulation-based inference

Alvaro Tejero-Cantero[e, 1], **Jan Boelts**[e, 1], **Michael Deistler**[e, 1], **Jan-Matthis Lueckmann**[e, 1], **Conor Durkan**[e, 2], **Pedro J. Gonçalves**[1, 3], **David S. Greenberg**[1, 4], **and Jakob H. Macke**[1, 5, 6]

The Journal of Open Source Software

Home | Search | mackelab/sbi 166 Stars · 35 Forks

sbi

- Home
- Installation
- Tutorials and Examples
- Contribute
- API Reference
- FAQ
- Credits

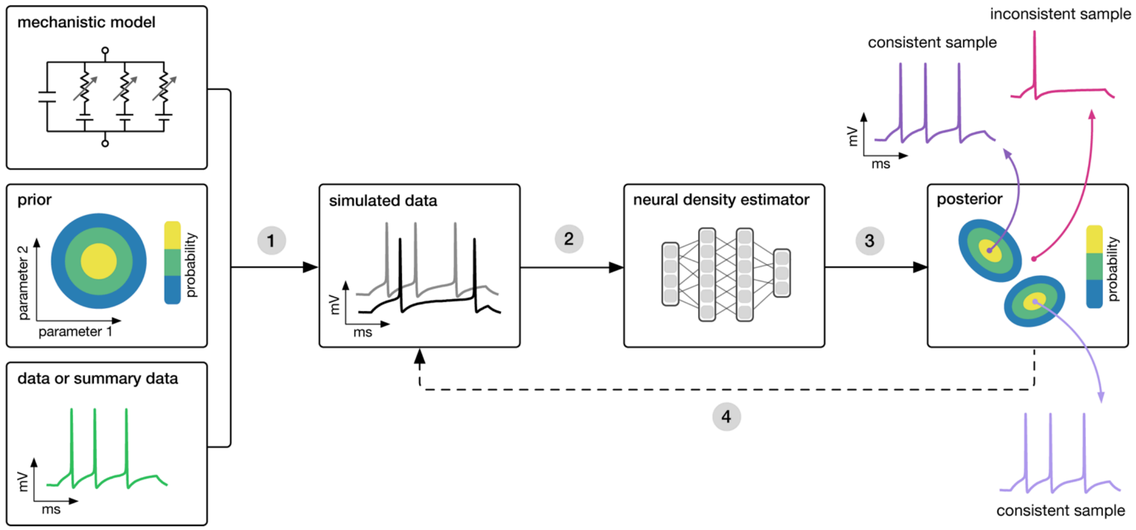
`sbi` implements three powerful machine-learning methods that address this problem:

- Sequential Neural Posterior Estimation (SNPE),
- Sequential Neural Likelihood Estimation (SNLE), and
- Sequential Neural Ratio Estimation (SNRE).

Depending on the characteristics of the problem, e.g. the dimensionalities of the parameter space and the observation space, one of the methods will be more suitable.

Table of contents

- Motivation and approach
- Publications
  - SNPE
  - SNLE
  - SNRE



Goal: Algorithmically identify mechanistic models which are consistent with data.

https://www.mackelab.org/sbi/

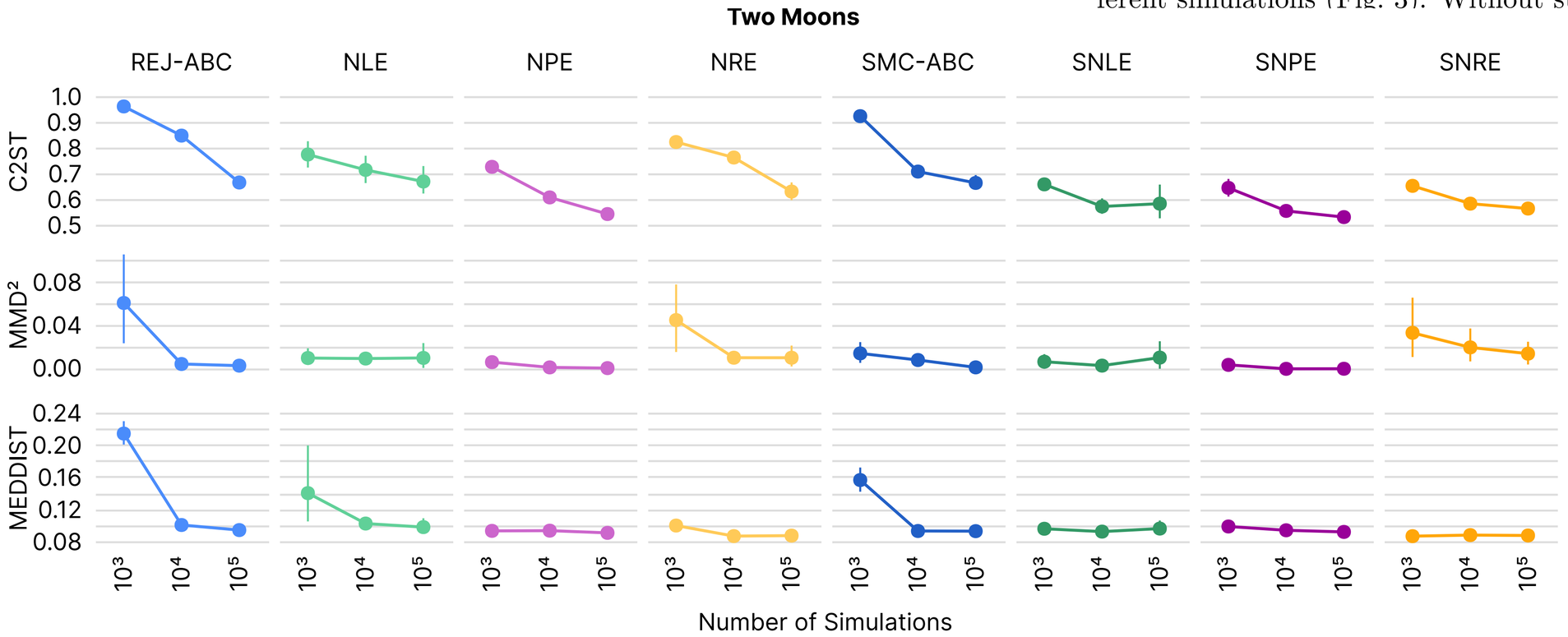
Benchmarking

Benchmarking Simulation-Based Inference

Jan-Matthis Lueckmann[1,4] Jan Boelts[1] David S. Greenberg[1,2]
Pedro J. Gonçalves[3] Jakob H. Macke[1,4,5]

#3: Sequential estimation improves sample efficiency. Our results show that sequential algorithms outperform non-sequential ones (Fig. 3). The difference was small on simple tasks (i.e. linear Gaussian cases), yet pronounced on most others. However, we also found these methods to exhibit diminishing returns as the simulation budget grows, which points to an opportunity for future improvements.

#4: Density or ratio estimation-based algorithms generally outperform classical techniques. REJ-ABC and SMC-ABC were generally outperformed by more recent techniques which use neural networks for density- or ratio-estimation, and which can therefore efficiently interpolate between different simulations (Fig. 3). Without such model-based

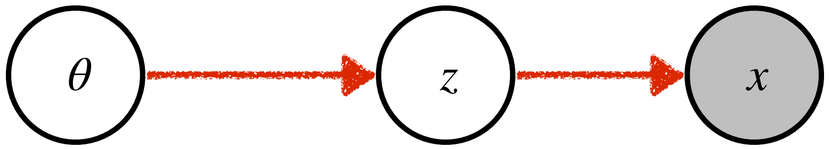


Single observation

When dealing with a single observation, there is a clear advantage to sequential techniques

- Use simulation budget in relevant regions of parameter space

#3: Sequential estimation improves sample efficiency. Our results show that sequential algorithms outperform non-sequential ones (Fig. 3). The difference was small on simple tasks (i.e. linear Gaussian cases), yet pronounced on most others. However, we also found these methods to exhibit diminishing returns as the simulation budget grows, which points to an opportunity for future improvements.



$$p(x, z \mid \theta) = p(x \mid z)p(z \mid \theta)$$

$$p(\theta \mid x) \propto \int \underbrace{p(\theta)}_{\text{prior}} \underbrace{p(x, z \mid \theta)}_{\text{likelihood}} dz$$

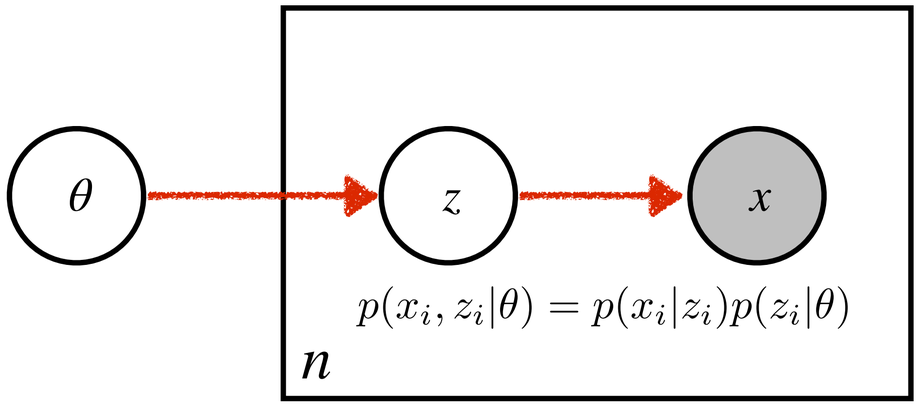
$$p(x \mid \theta) = \int p(x, z \mid \theta) dz$$

$$p(\theta \mid x) \propto \underbrace{p(\theta)}_{\text{prior}} \underbrace{p(x \mid \theta)}_{\text{likelihood}}$$

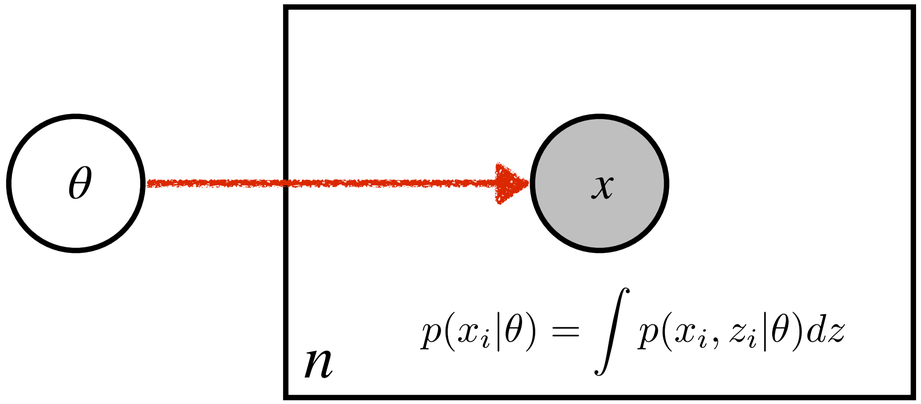
iid data and amortized likelihood

However, when dealing with iid data, there is a more advantage to learning an amortized likelihood ratio that is accurate everywhere and can be reused.

- More work needed to study tradeoff of sequential approaches with iid data



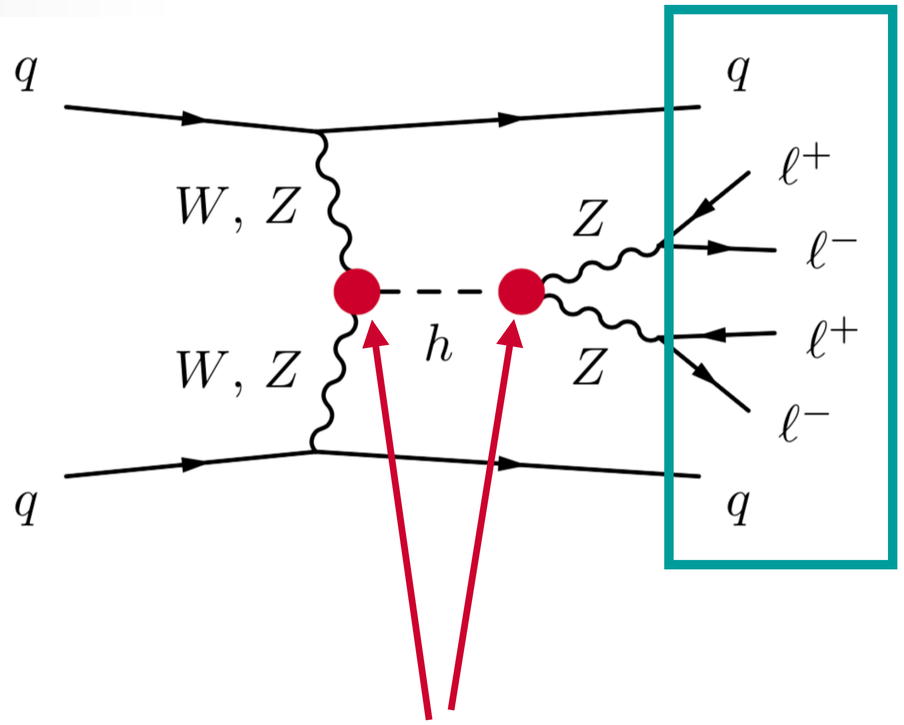
$$p(\theta \mid \{x_i\}) \propto \underbrace{p(\theta)}_{\text{prior}} \prod_{i=1}^{n} \left[\int \underbrace{p(x_i, z_i \mid \theta)}_{\text{joint likelihood}} \, dz_i \right]$$



$$p(\theta \mid \{x_i\}) \propto \underbrace{p(\theta)}_{\text{prior}} \prod_{i=1}^{n} \left[\underbrace{p(x_i \mid \theta)}_{\text{amortized likelihood}} \right]$$

Examples

Impact on Studies of The Higgs Boson



42-Dim observable **x**

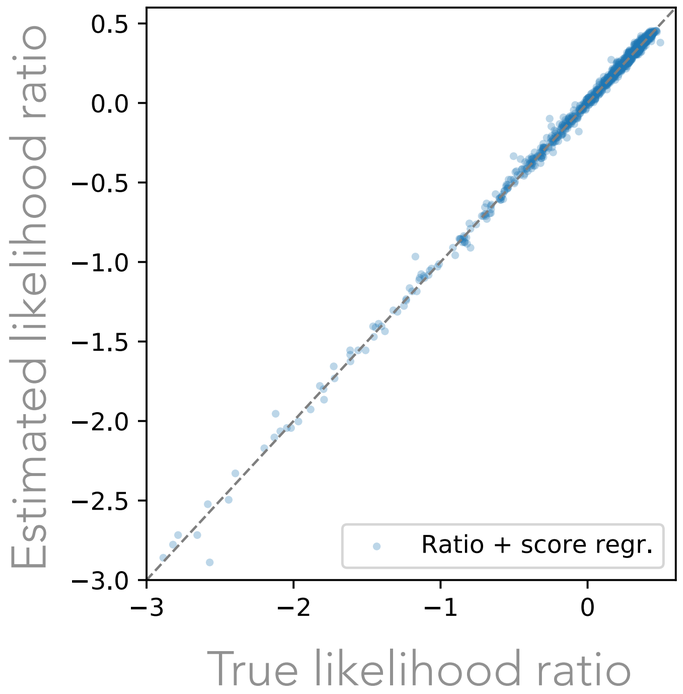
Exciting new physics might hide here!
We parameterize it with two coefficients:

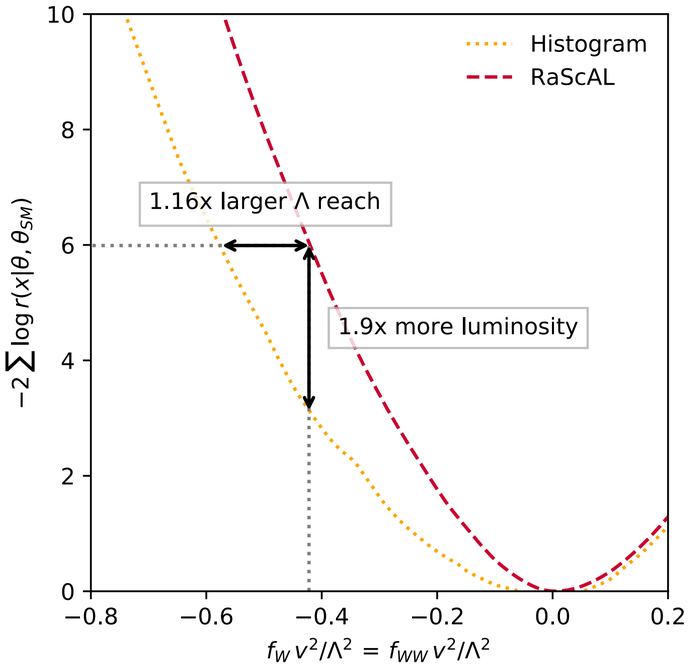
$$\mathcal{L} = \mathcal{L}_{\rm SM} + \boxed{\frac{f_W}{\Lambda^2}} \underbrace{\frac{ig}{2}\,(D^\mu\phi)^\dagger\,\sigma^a\,D^\nu\phi\,W^a_{\mu\nu}}_{\mathcal{O}_W} - \boxed{\frac{f_{WW}}{\Lambda^2}} \underbrace{\frac{g^2}{4}\,(\phi^\dagger\phi)\,W^a_{\mu\nu}\,W^{\mu\nu\,a}}_{\mathcal{O}_{WW}}$$

J Brehmer, J Pavez, G Louppe, K.C. PRL & PRD 2018 [arXiv:1805.00013 & arXiv:1805.00020], CARL [arxiv:1506.02169]

Impact on Studies of The Higgs Boson

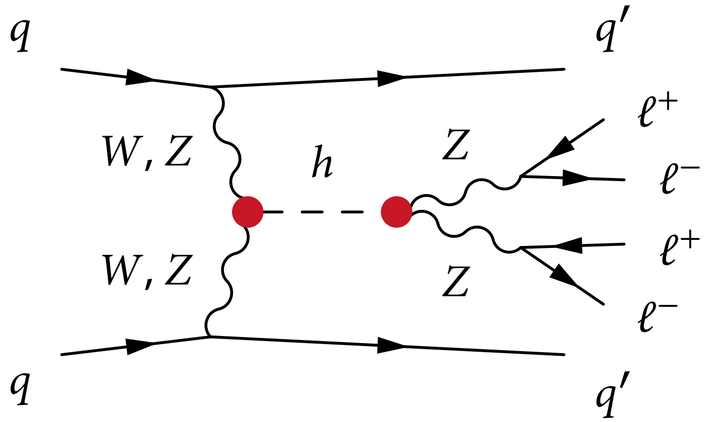
(based on a 42-Dim observation $\boldsymbol{x}$)





Accurate likelihood ratio estimates without the need for summary statistics improves sensitivity significantly

- Equivalent to 90% more LHC data!

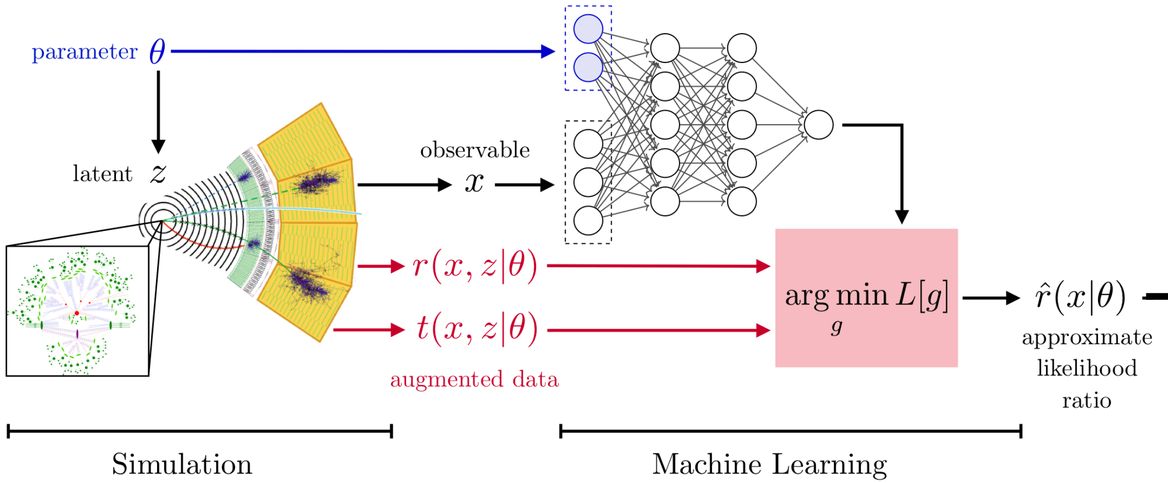


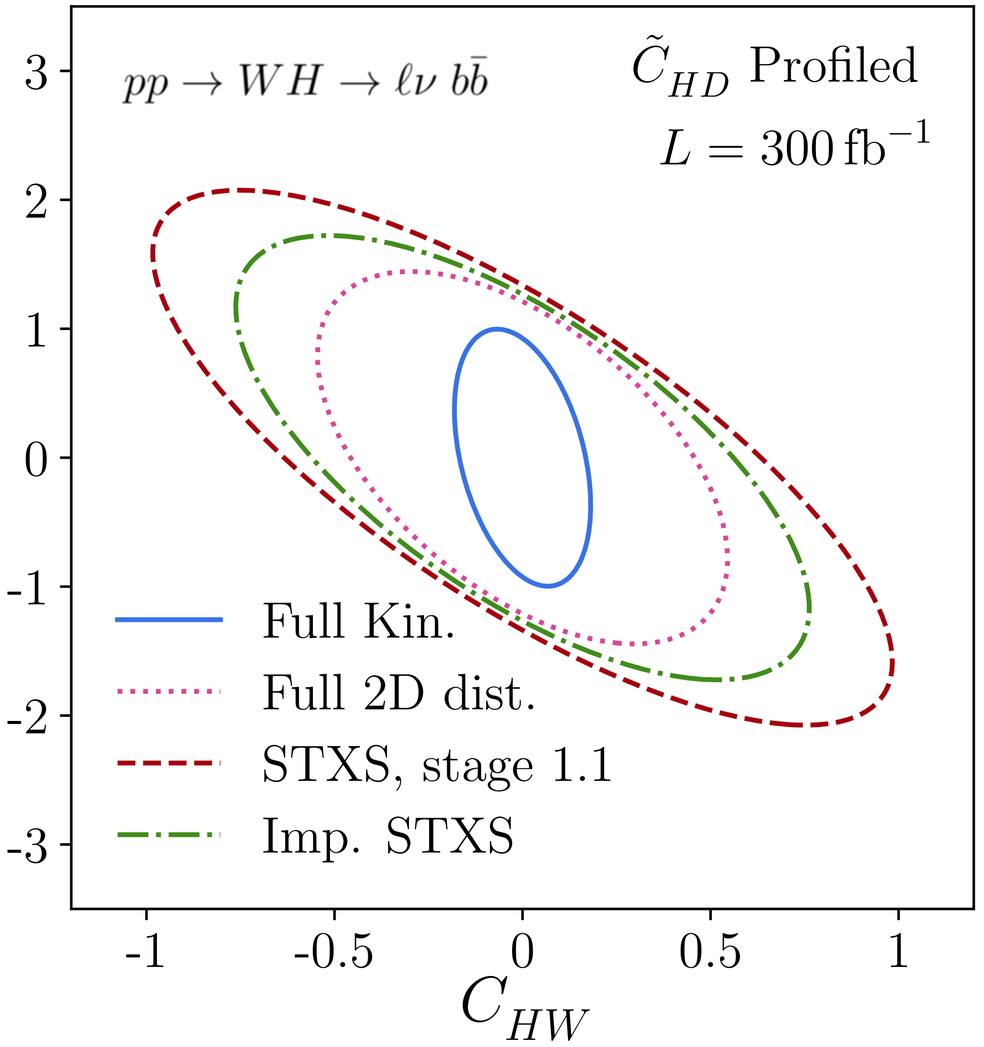
J Brehmer, J Pavez, G Louppe, K.C. PRL & PRD 2018 [arXiv:1805.00013 & arXiv:1805.00020], CARL [arxiv:1506.02169]

Impact on Studies of The Higgs Boson

Massive gains in precision of a flagship measurement at the LHC !

Equivalent increasing data collected by LHC by several factors





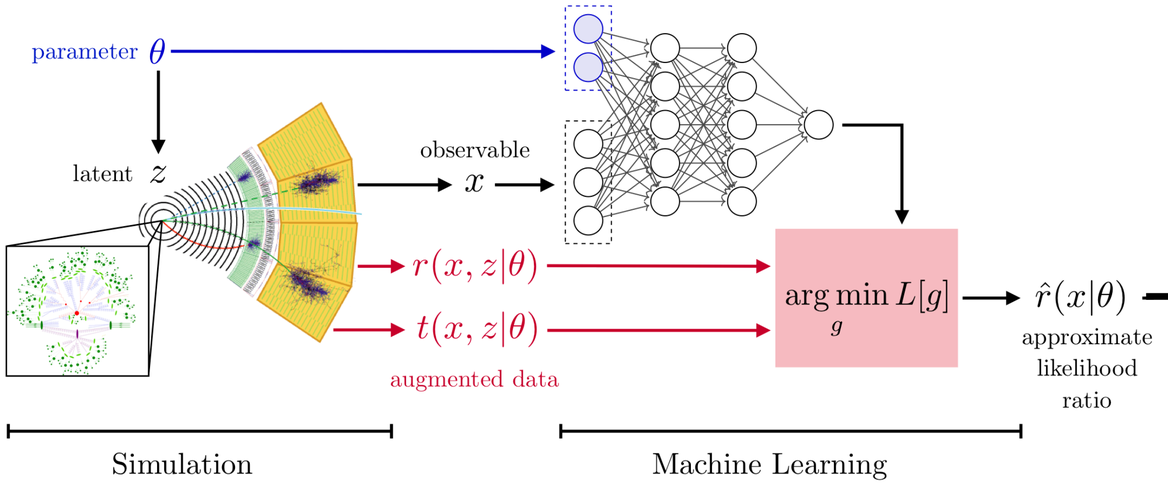
[J. Brehmer, S. Dawson, S. Homiller, F. Kling, T. Plehn 1908.06980]

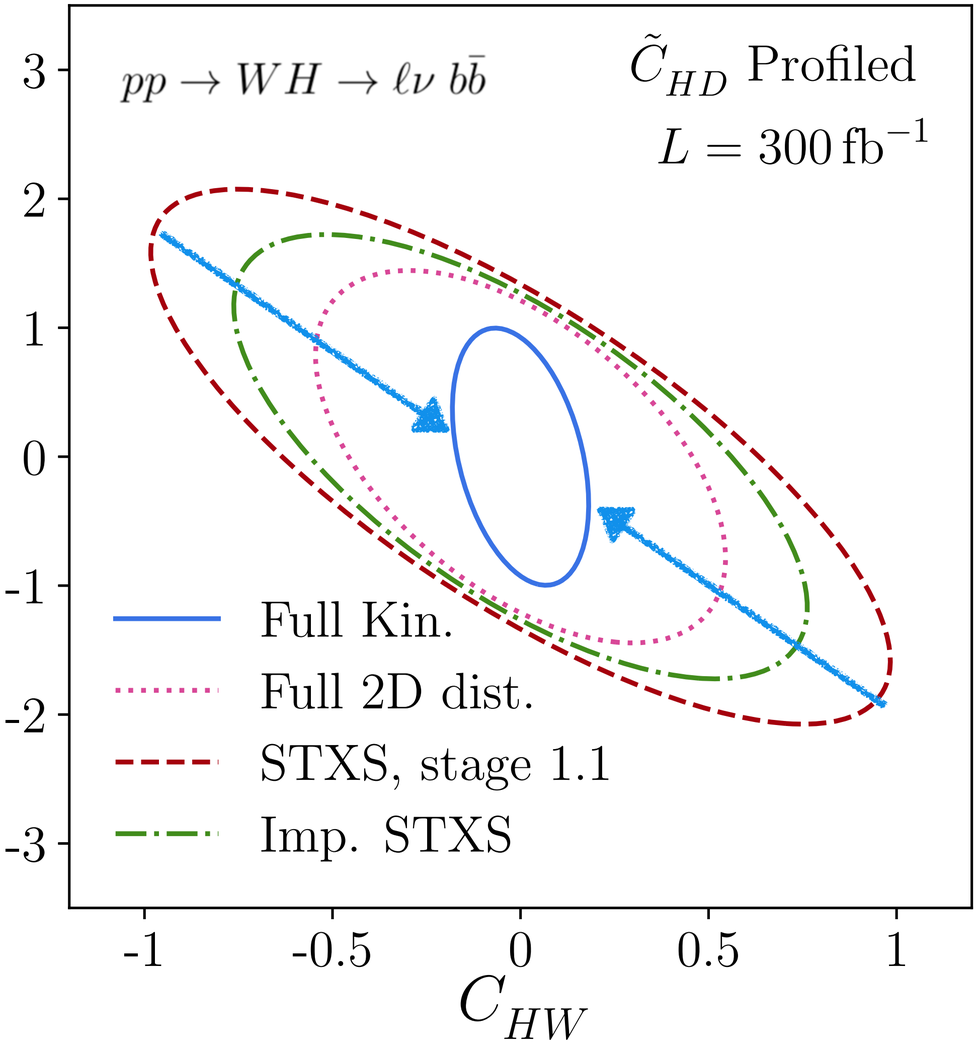
[J. Brehmer, F. Kling, I. Espejo, K. Cranmer 1907.10621]

Impact on Studies of The Higgs Boson

Massive gains in precision of a flagship measurement at the LHC !

Equivalent increasing data collected by LHC by several factors



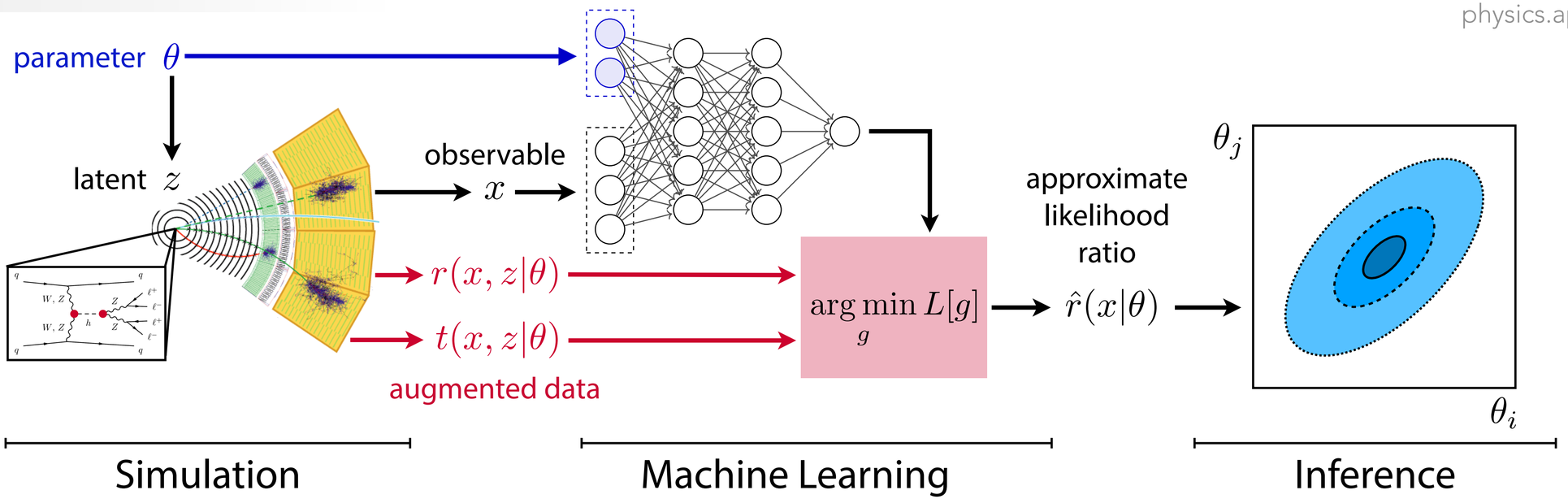


[J. Brehmer, S. Dawson, S. Homiller, F. Kling, T. Plehn 1908.06980]

[J. Brehmer, F. Kling, I. Espejo, K. Cranmer 1907.10621]

Learning the likelihood ratio

arXiv:1805.12244
PRL, arXiv:1805.00013
PRD, arXiv:1805.00020
physics.aps.org/articles/v11/90



MadMiner: Machine learning–based inference for particle physics

By Johann Brehmer, Felix Kling, Irina Espejo, and Kyle Cranmer

pypi package 0.6.3 | build passing | docs failing | chat on gitter | code style black | License MIT | DOI 10.5281/zenodo.1489147 | arXiv 1907.10621

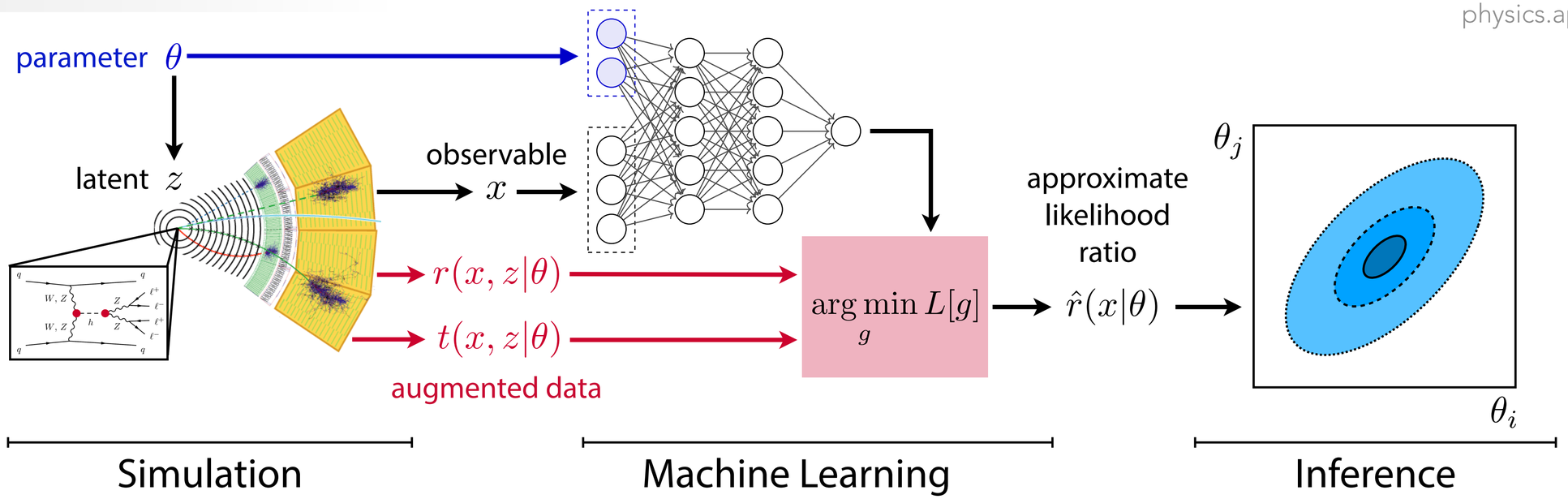
Introduction

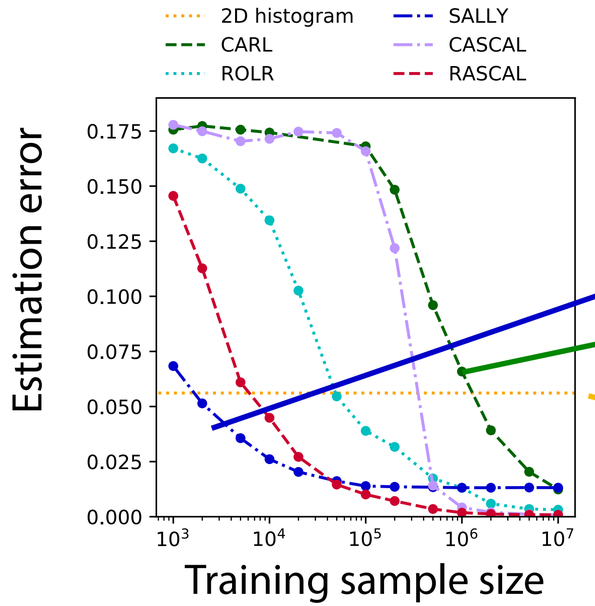
Particle physics processes are usually modeled with complex Monte-Carlo simulations of the hard process, parton shower, and detector interactions. These simulators typically do not admit a tractable likelihood function: given a (potentially high-dimensional) set of observables, it is usually not possible to calculate the probability of these observables for some model parameters. Particle physicisists usually tackle this problem of "likelihood-free inference" by hand-picking a few "good" observables or summary statistics and filling histograms of them. But this conventional

Dedicated software package interfacing with particle physics simulators:
github.com/johannbrehmer/madminer

Learning the likelihood ratio

arXiv:1805.12244
PRL, arXiv:1805.00013
PRD, arXiv:1805.00020
physics.aps.org/articles/v11/90





New techniques
require less data than
without augmented data

Traditional Approach no NN

Computing challenges

Created REANA workflows for MadMiner

- Containers for simulation & ML tasks
- Would like GPU-accelerated nodes for ML tasks

reana

Reproducible research data analysis platform

Flexible

Run many computational workflow engines.

Scalable

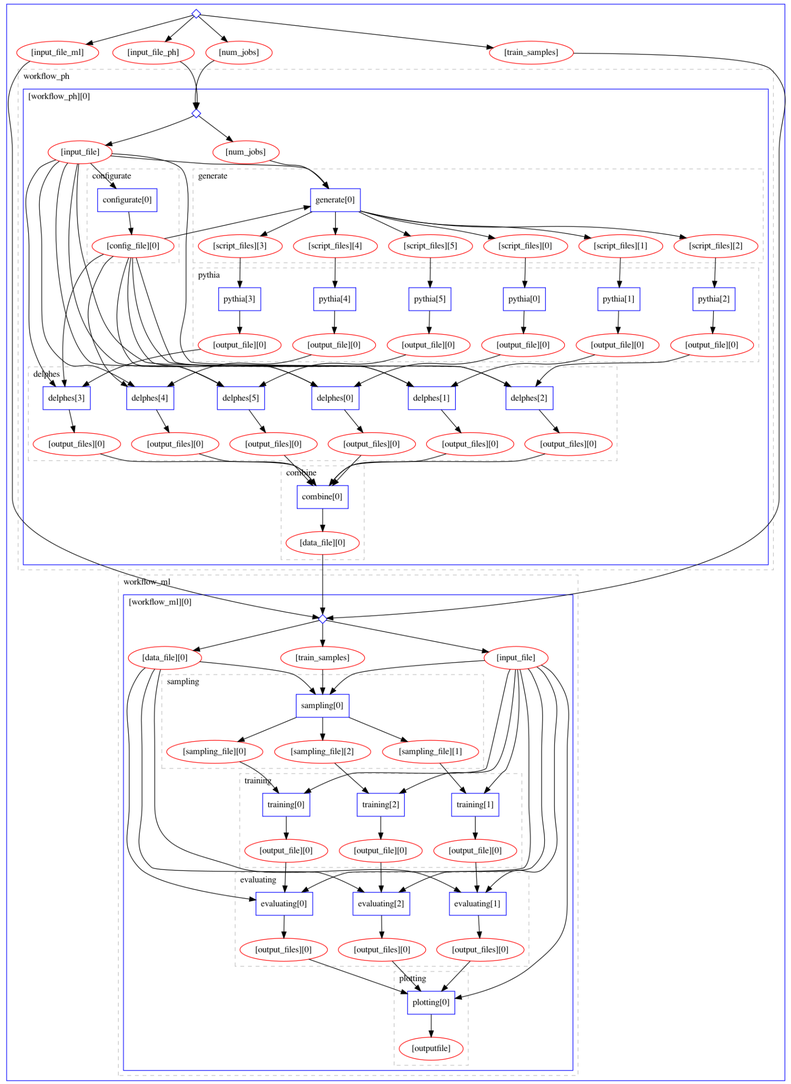
Support for remote compute clouds.

Reusable

Containerise once, reuse elsewhere. Cloud-native.

Free

Free Software. MIT licence. Made with ❤ at CERN.



Computing challenges

Created REANA workflows for MadMiner

- Containers for simulation & ML tasks
- Would like GPU-accelerated nodes for ML tasks

reana

Reproducible research data analysis platform

Flexible

Run many computational workflow engines.

Scalable

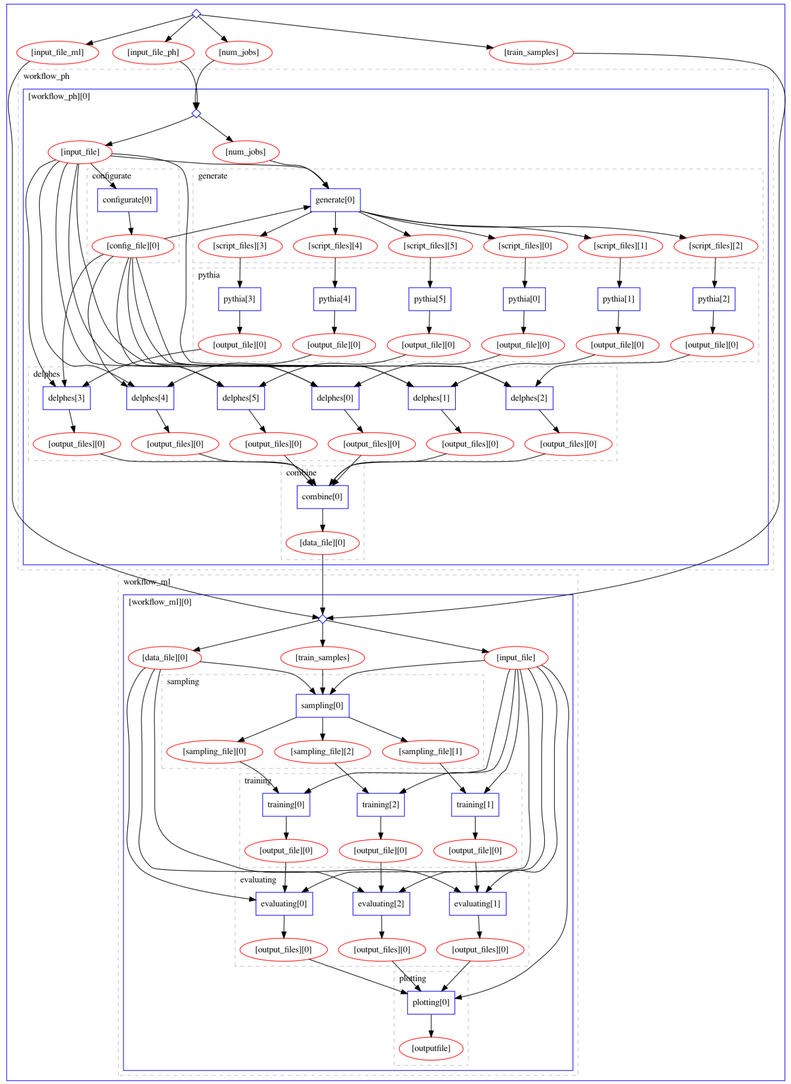
Support for remote compute clouds.

Reusable

Containerise once, reuse elsewhere. Cloud-native.

Free

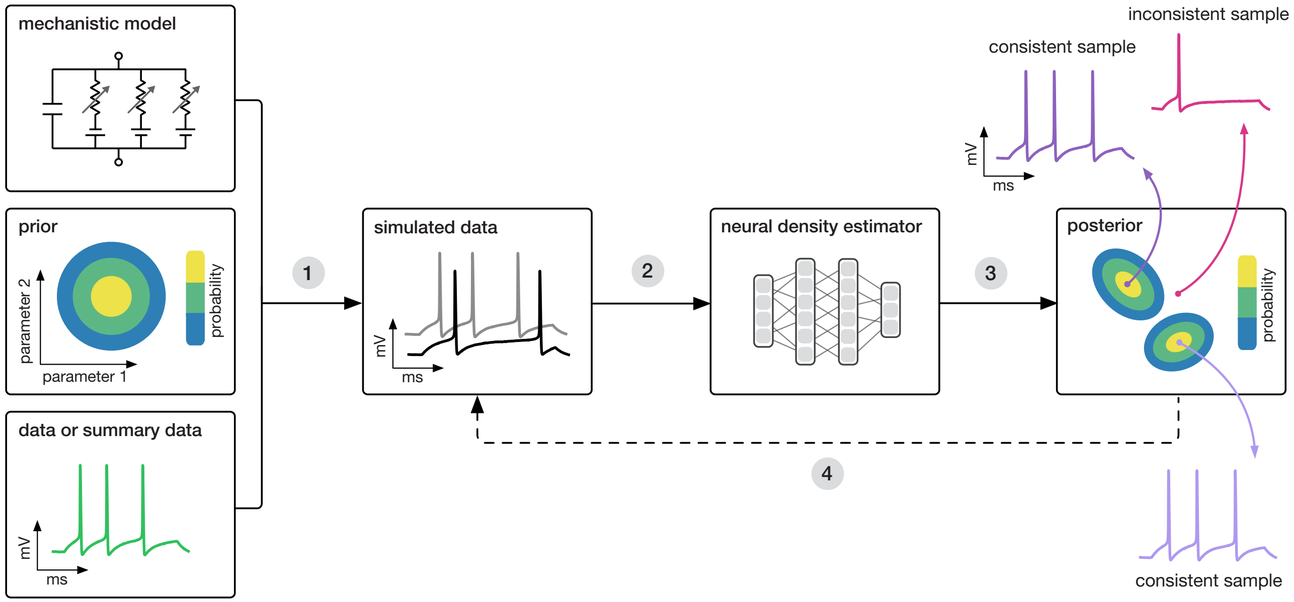
Free Software. MIT licence. Made with ❤ at CERN.



From Jakob Macke's group that developed the **sbi** python package

sbi: A toolkit for simulation-based inference

Alvaro Tejero-Cantero[e, 1], Jan Boelts[e, 1], Michael Deistler[e, 1], Jan-Matthis Lueckmann[e, 1], Conor Durkan[e, 2], Pedro J. Gonçalves[1, 3], David S. Greenberg[1, 4], and Jakob H. Macke[1, 5, 6]



eLife

Training deep neural density estimators to identify mechanistic models of neural dynamics

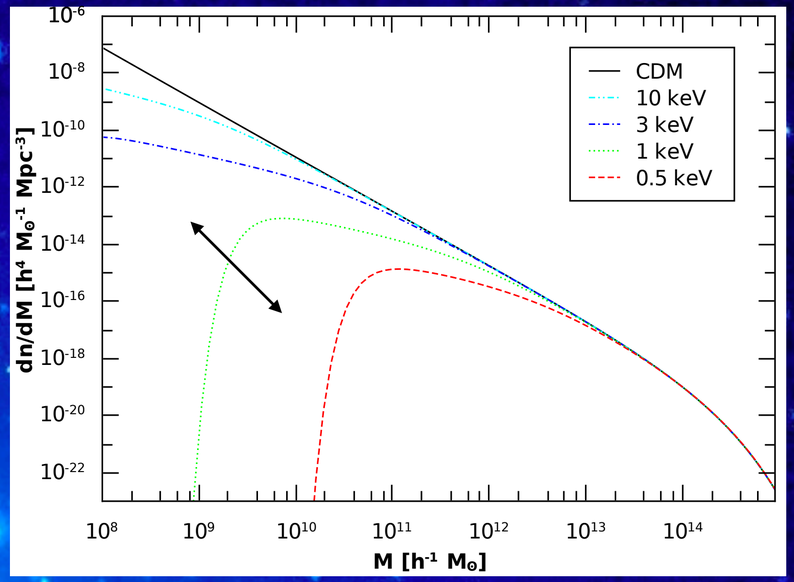
Pedro J Gonçalves[1,2†*], Jan-Matthis Lueckmann[1,2†*], Michael Deistler[1,3†*], Marcel Nonnenmacher[1,2,4], Kaan Öcal[2,5], Giacomo Bassetto[1,2], Chaitanya Chintaluri[6,7], William F Podlaski[6], Sara A Haddad[8], Tim P Vogels[6,7], David S Greenberg[1,4], Jakob H Macke[1,2,3,9*]

[1]Computational Neuroengineering, Department of Electrical and Computer Engineering, Technical University of Munich, Munich, Germany; [2]Max Planck Research Group Neural Systems Analysis, Center of Advanced European Studies and Research (caesar), Bonn, Germany; [3]Machine Learning in Science, Excellence Cluster Machine Learning, Tübingen University, Tübingen, Germany; [4]Model-Driven Machine Learning, Institute of Coastal Research, Helmholtz Centre Geesthacht, Geesthacht, Germany; [5]Mathematical Institute, University of Bonn, Bonn, Germany; [6]Centre for Neural Circuits and Behaviour, University of Oxford, Oxford, United Kingdom; [7]Institute of Science and Technology Austria, Klosterneuburg, Austria; [8]Max Planck Institute for Brain Research, Frankfurt, Germany; [9]Max Planck Institute for Intelligent Systems, Tübingen, Germany

Abstract Mechanistic modeling in neuroscience aims to explain observed phenomena in terms of underlying causes. However, determining which model parameters agree with complex and stochastic neural data presents a significant challenge. We address this challenge with a machine learning tool which uses deep neural density estimators—trained using model simulations—to carry out Bayesian inference and retrieve the full space of parameters compatible with raw data or selected data features. Our method is scalable in parameters and data features and can rapidly analyze new data after initial training. We demonstrate the power and flexibility of our approach on receptive fields, ion channels, and Hodgkin–Huxley models. We also characterize the space of circuit configurations giving rise to rhythmic activity in the crustacean stomatogastric ganglion, and use these results to derive hypotheses for underlying compensation mechanisms. Our approach will help close the gap between data-driven and theory-driven models of neural dynamics.

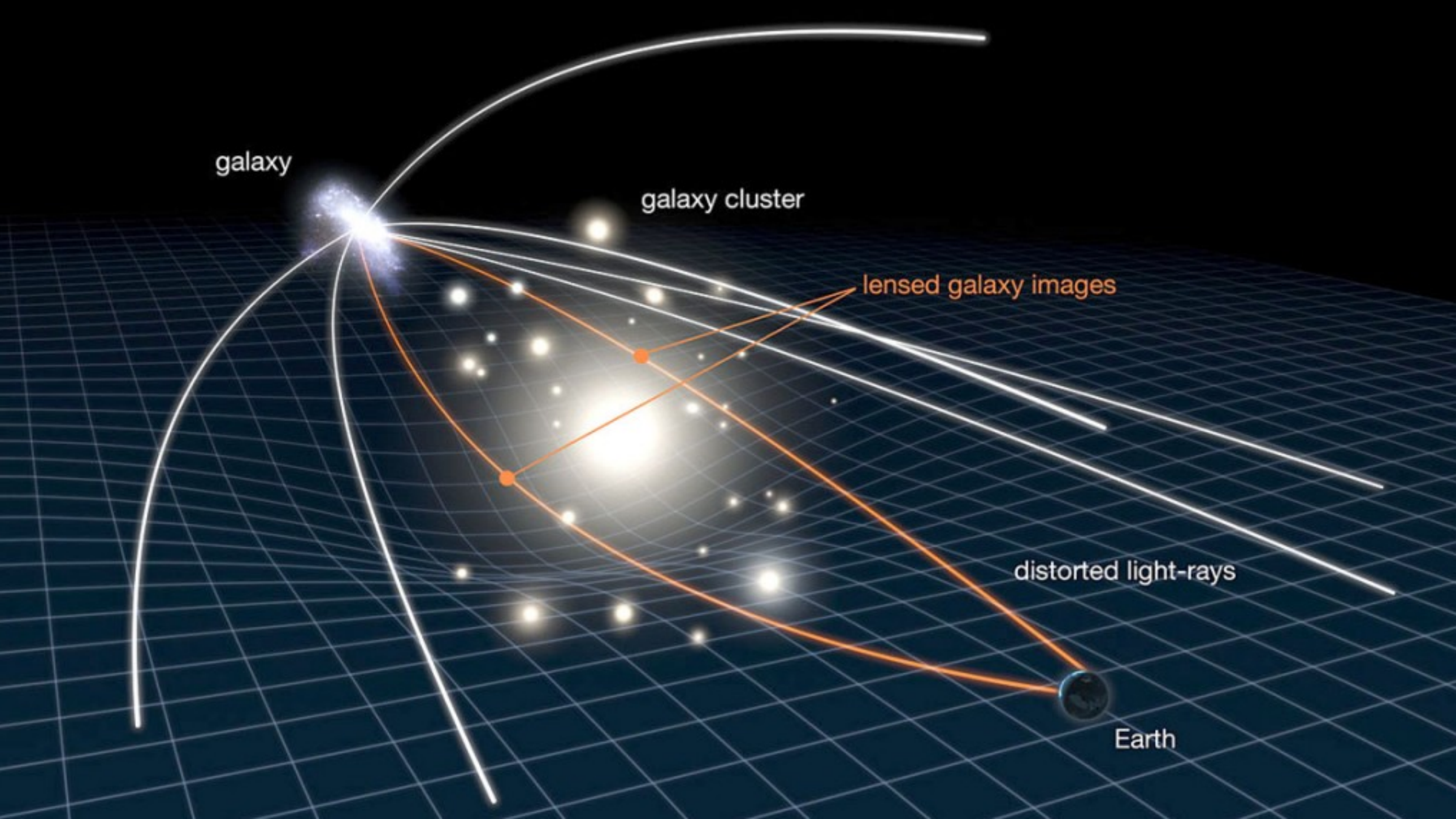
Dark Matter Substructure

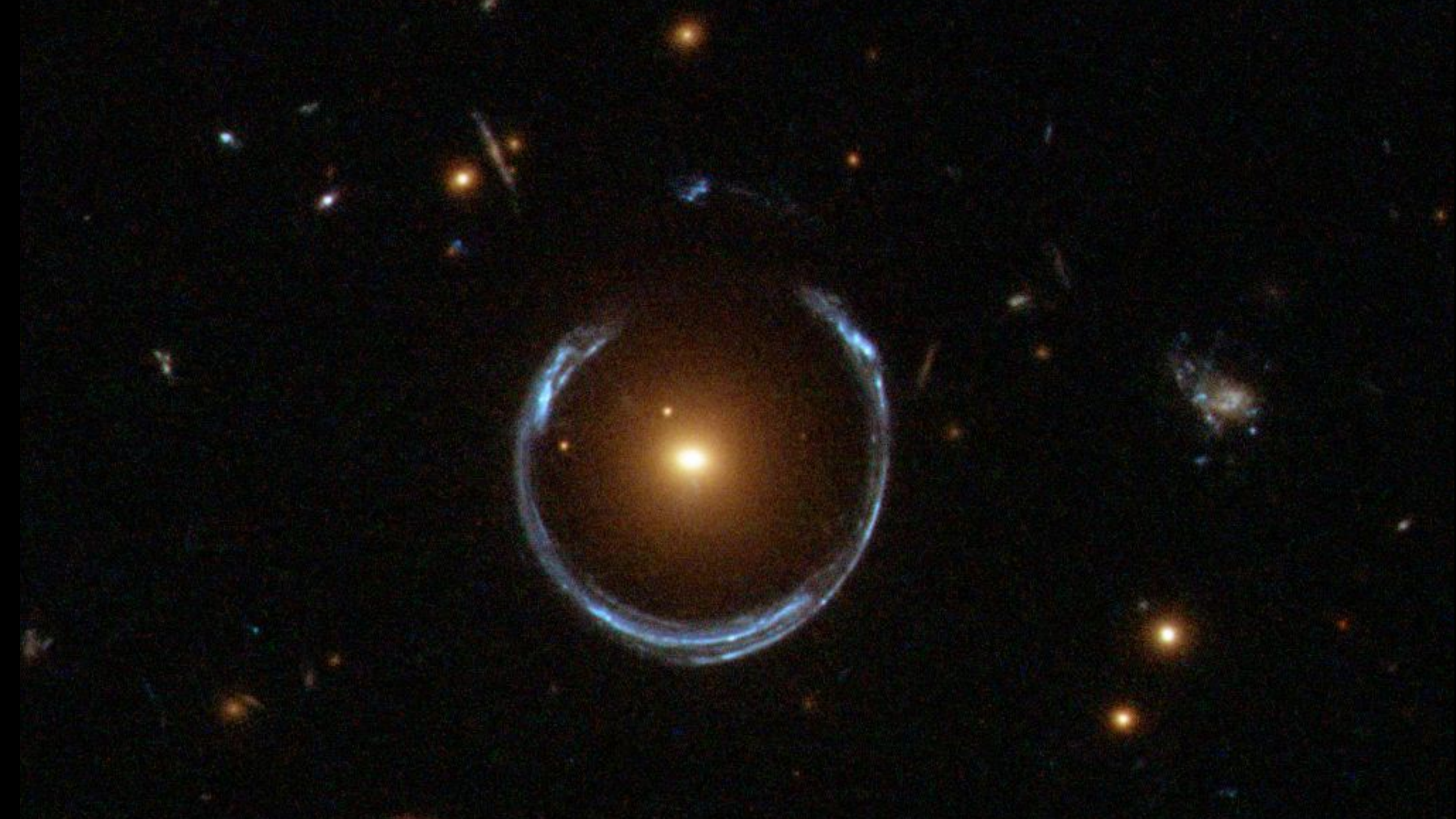
Abundance of DM subhalos vs mass:



[R. *Dunstan et al* 1109.6291]

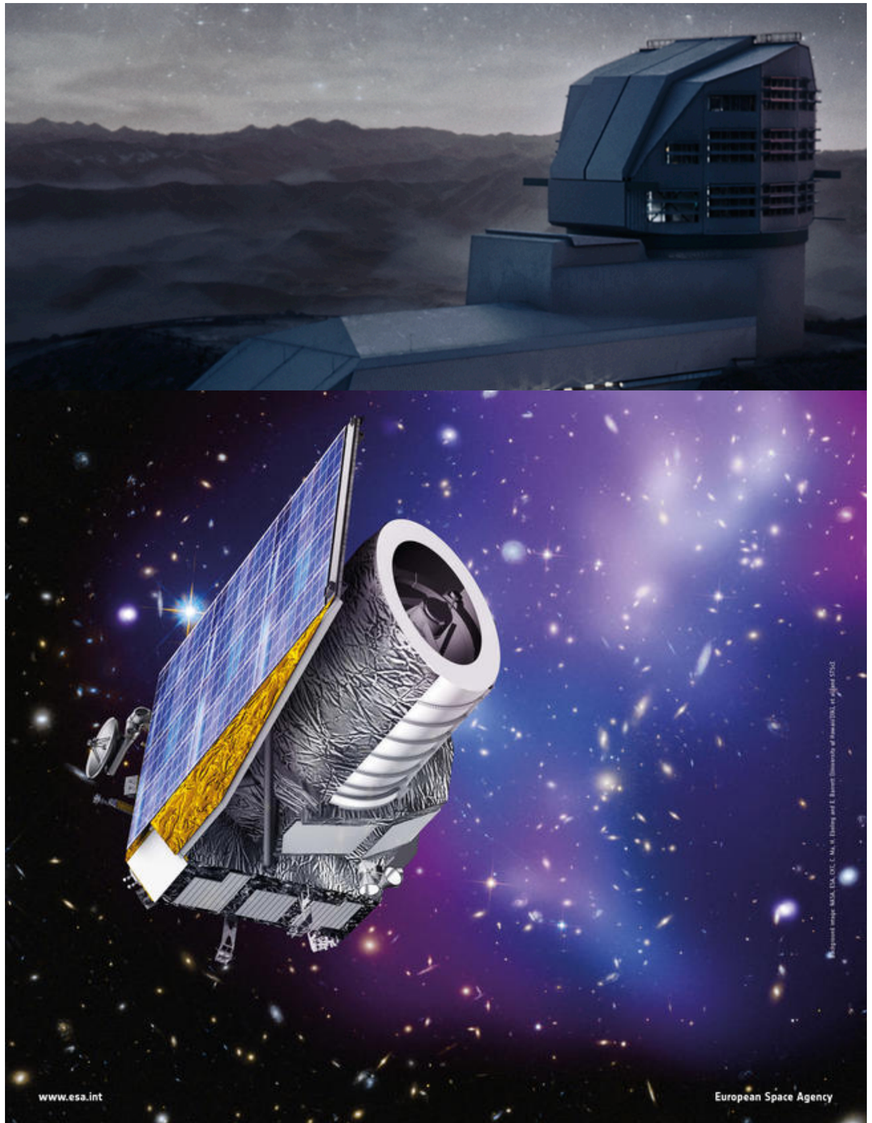
[T. Brown, J.Tumlinson]

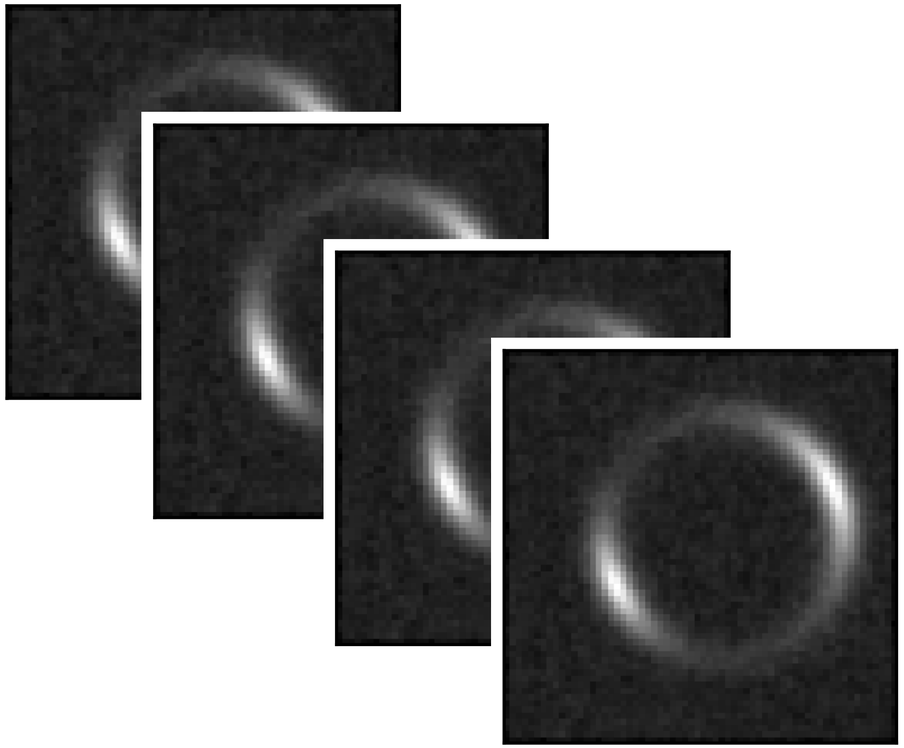

galaxy
galaxy cluster
lensed galaxy images
distorted light-rays
Earth



Scalable inference for small subhalos

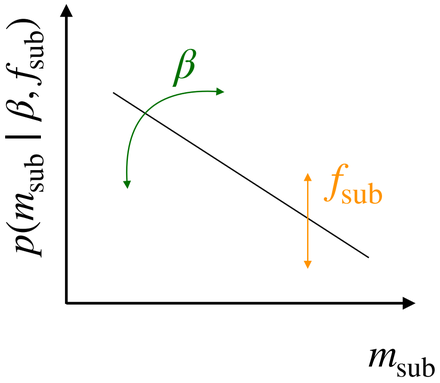
Future surveys (LSST, Euclid) are expected to deliver large samples of galaxy-galaxy strong lenses [Collett et al 1507.02657]





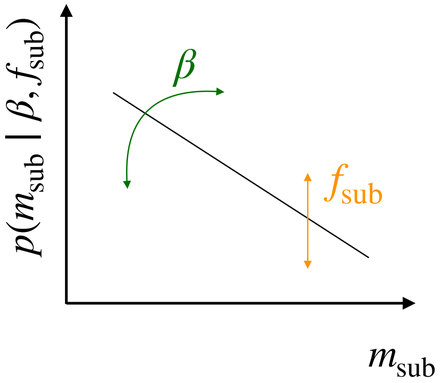
Simulation-based inference for strong lensing

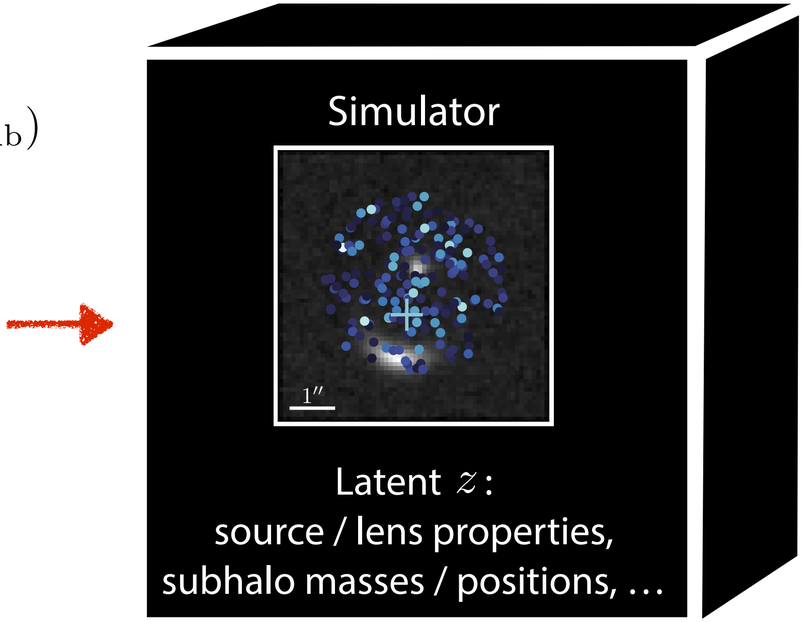
2 parameters $\theta = (\beta, f_{\rm sub})$



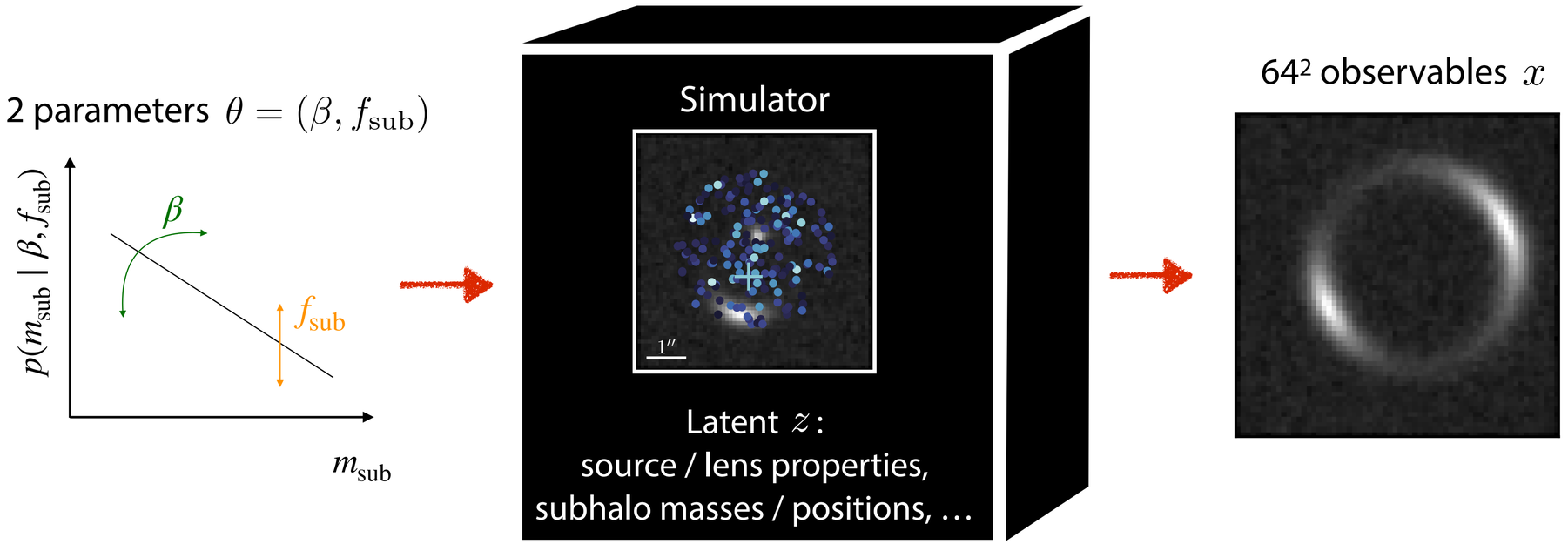
Simulation-based inference for strong lensing

2 parameters $\theta = (\beta, f_{\rm sub})$





Simulation-based inference for strong lensing

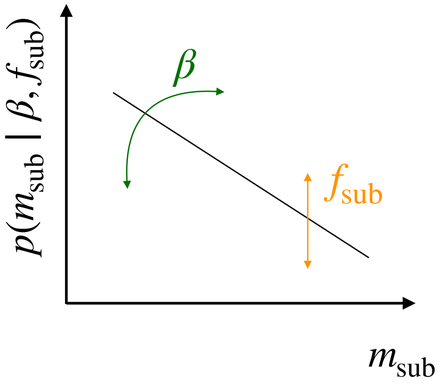


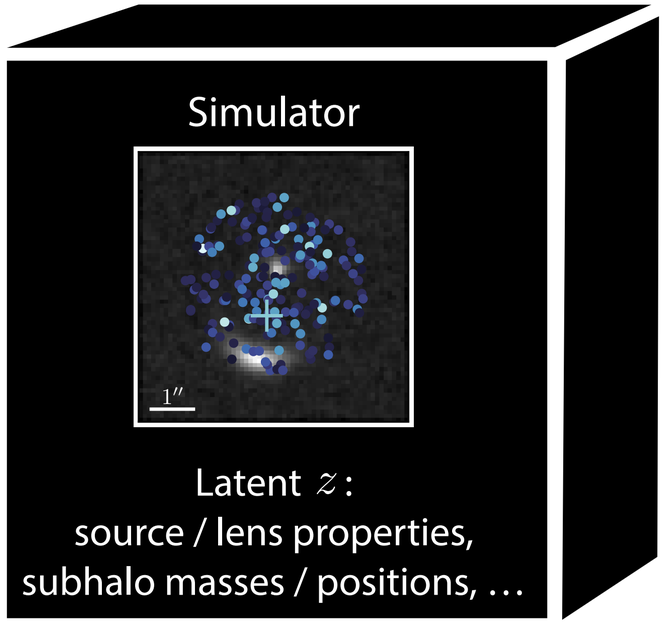
⇒ Need inference technique that

- scales to many lenses (fast evaluation)
- captures subtle effects in high-dimensional image data
- can deal with a large number of subhalos (latent variables)

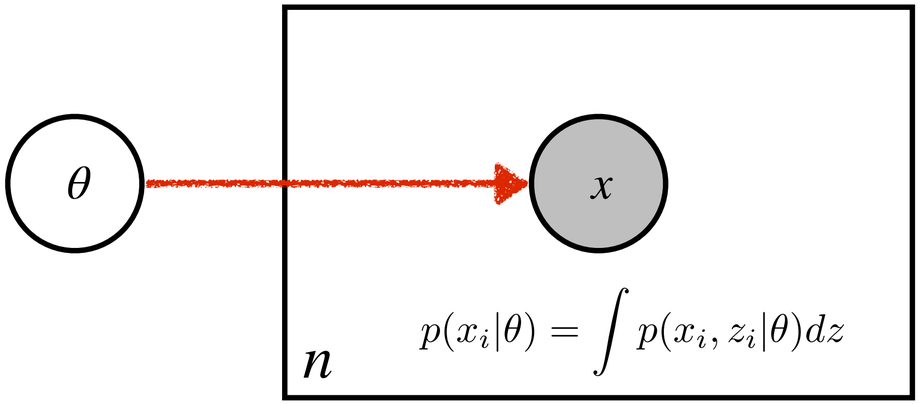
Simulation-based inference for strong lensing

2 parameters $\theta = (\beta, f_{\rm sub})$





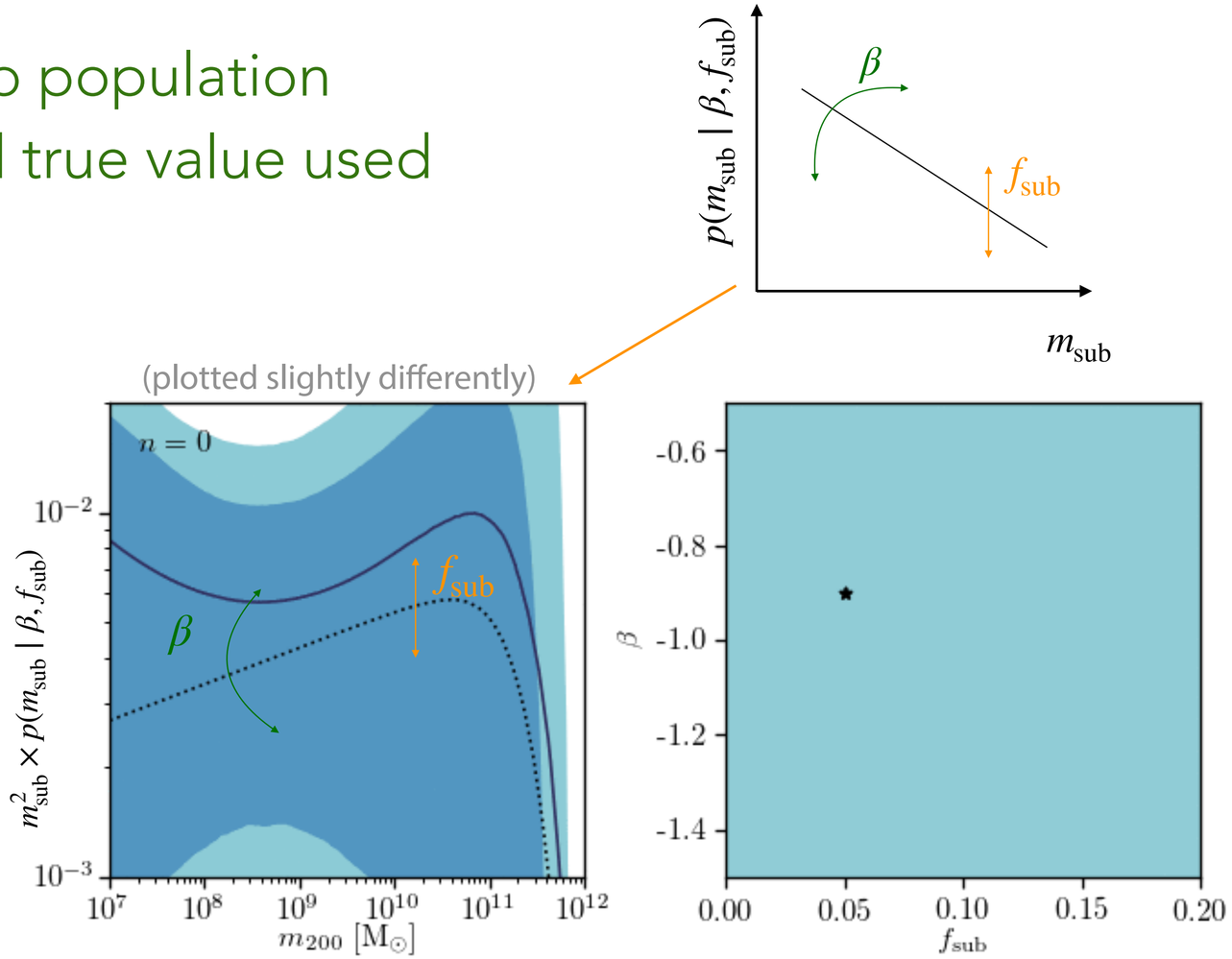
64^2 observables x



$$p(\theta \mid \{x_i\}) \propto \underbrace{p(\theta)}_{\text{prior}} \prod_{i=1}^{n} \left[\underbrace{p(x_i \mid \theta)}_{\text{amortized likelihood}} \right]$$

Posterior from amortized likelihood ratio

Watch how the posterior for two population parameters concentrate around true value used to generate mock data.




Sid Mishra-Sharma

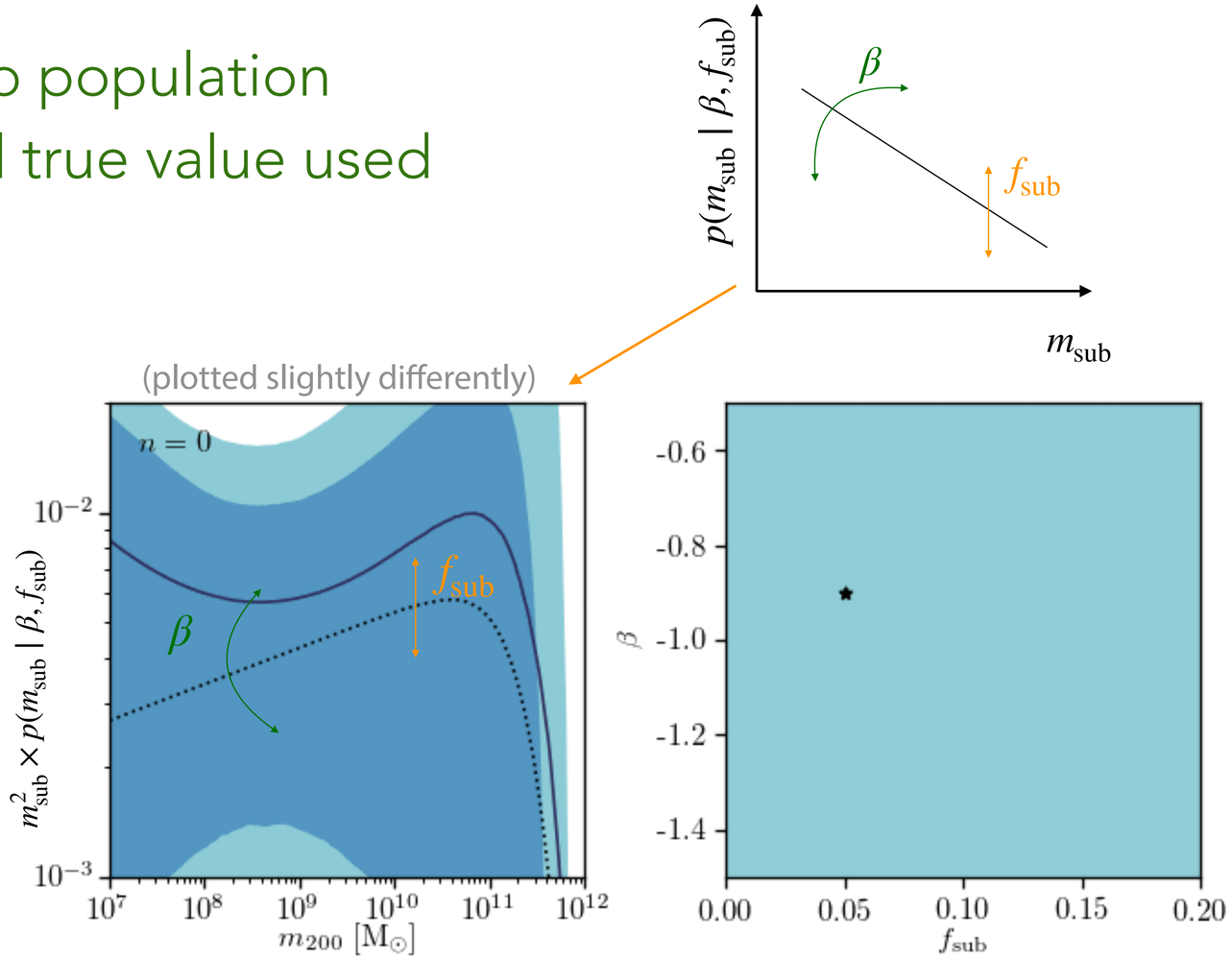

Johann Brehmer


Gilles Louppe


Joeri Hermans, K. Cranmer. published in ApJ, 886 (2019) arXiv:1909.02005

Posterior from amortized likelihood ratio

Watch how the posterior for two population parameters concentrate around true value used to generate mock data.



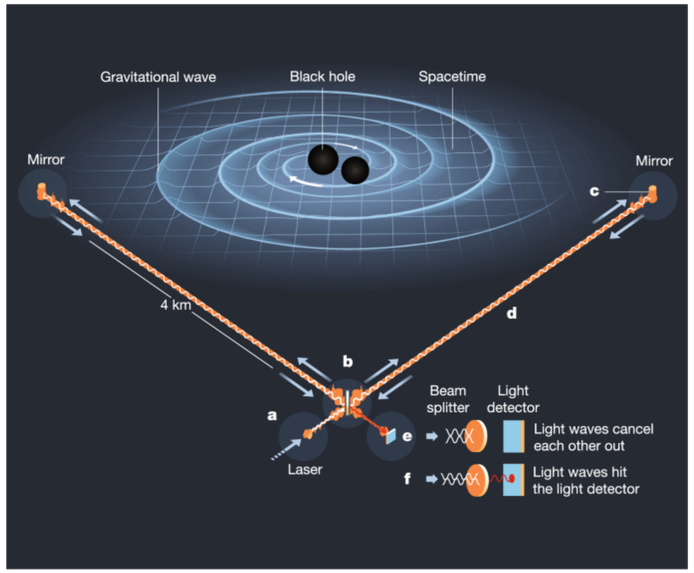

Sid Mishra-Sharma

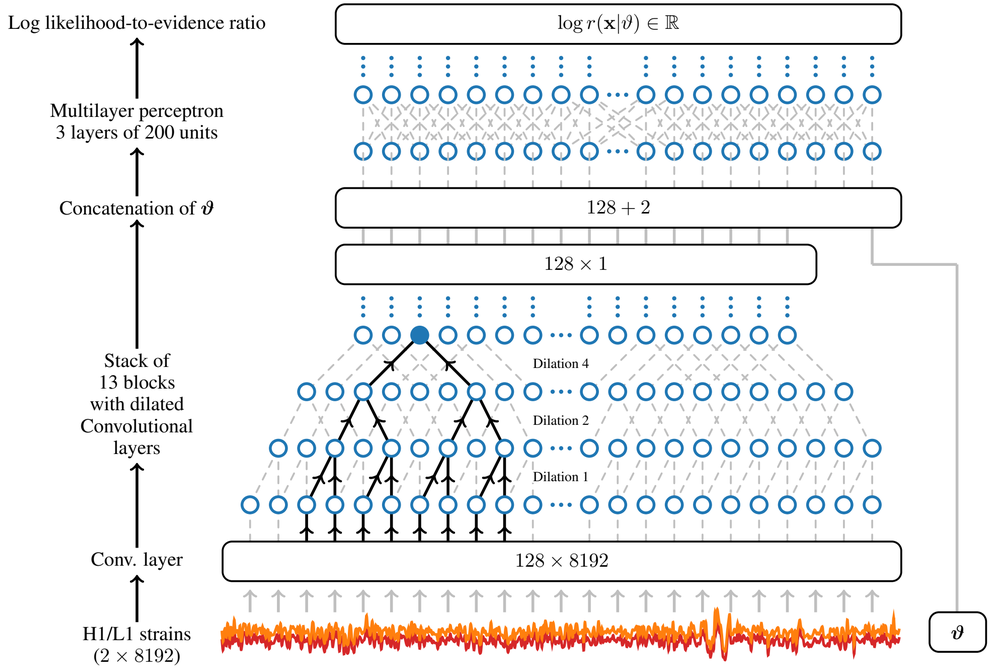

Johann Brehmer


Gilles Louppe


Joeri Hermans, K. Cranmer. published in ApJ, 886 (2019) arXiv:1909.02005

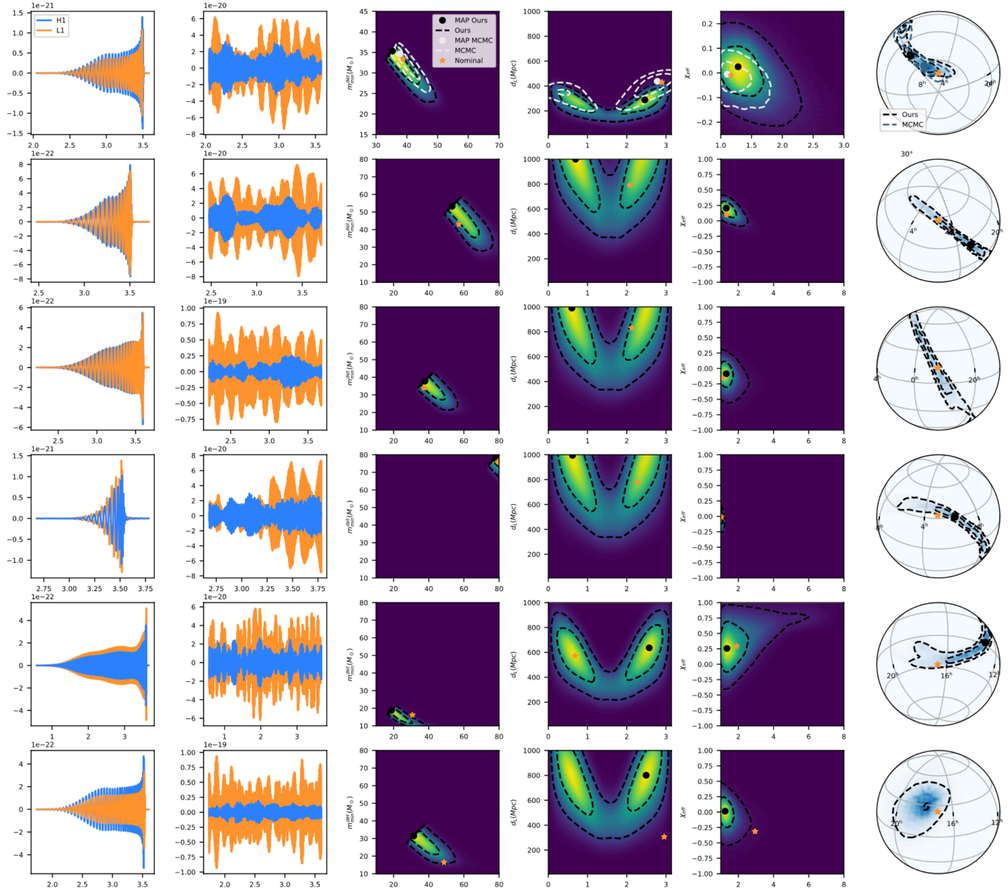
Gravitational Wave Astronomy



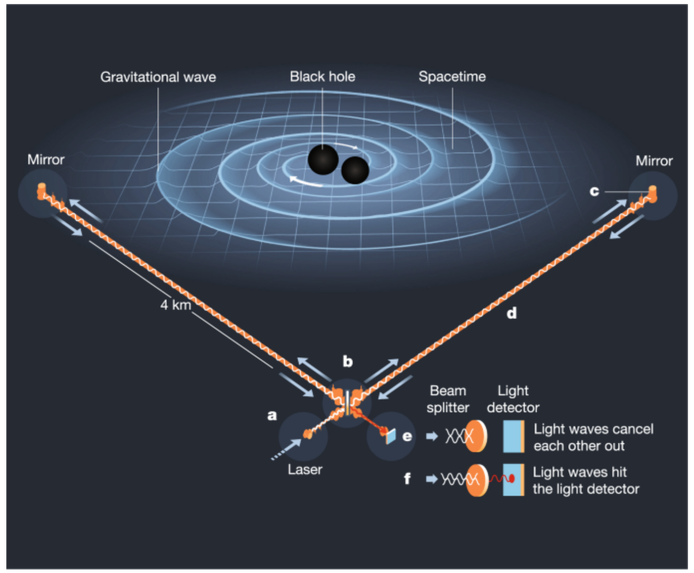


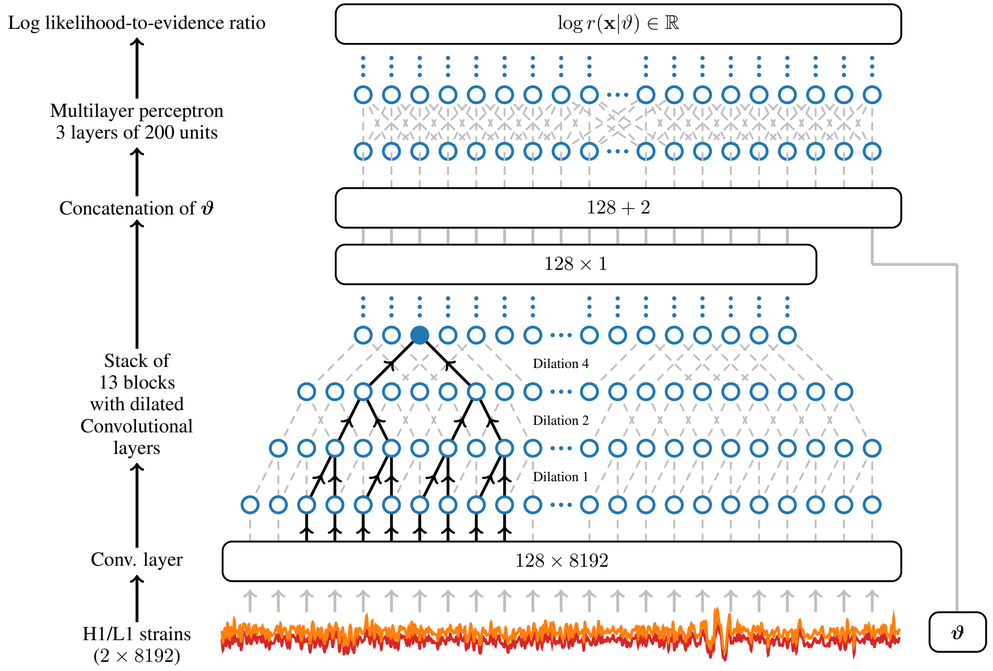
Lightning-Fast Gravitational Wave Parameter Inference through Neural Amortization

Delaunoy, Wehenkel, Hinderer, Nissanke, Weniger, Williamson, Louppe
[arXiv:2010.12931]



Gravitational Wave Astronomy





[28] H. Gabbard, C. Messenger, I. S. Heng, F. Tonolini, and R. Murray-Smith, Bayesian parameter estimation using conditional variational autoencoders for gravitational-wave astronomy (2019), arXiv:1909.06296 [astro-ph.IM].

[29] A. J. K. Chua and M. Vallisneri, Learning Bayesian posteriors with neural networks for gravitational-wave inference, Phys. Rev. Lett. **124**, 041102 (2020), arXiv:1909.05966 [gr-qc].

[30] C. Chatterjee, L. Wen, K. Vinsen, M. Kovalam, and A. Datta, Using Deep Learning to Localize Gravitational Wave Sources, Phys. Rev. D **100**, 103025 (2019), arXiv:1909.06367 [astro-ph.IM].

[31] S. R. Green, C. Simpson, and J. Gair, Gravitational-wave parameter estimation with autoregressive neural network flows, Phys. Rev. D **102**, 104057 (2020), arXiv:2002.07656 [astro-ph.IM].

[32] S. R. Green and J. Gair, Complete parameter inference for GW150914 using deep learning, Mach. Learn. Sci. Tech. **2**, 03LT01 (2021), arXiv:2008.03312 [astro-ph.IM].

[33] A. Delaunoy, A. Wehenkel, T. Hinderer, S. Nissanke, C. Weniger, A. R. Williamson, and G. Louppe, Lightning-Fast Gravitational Wave Parameter Inference through Neural Amortization, (2020), arXiv:2010.12931 [astro-ph.IM].

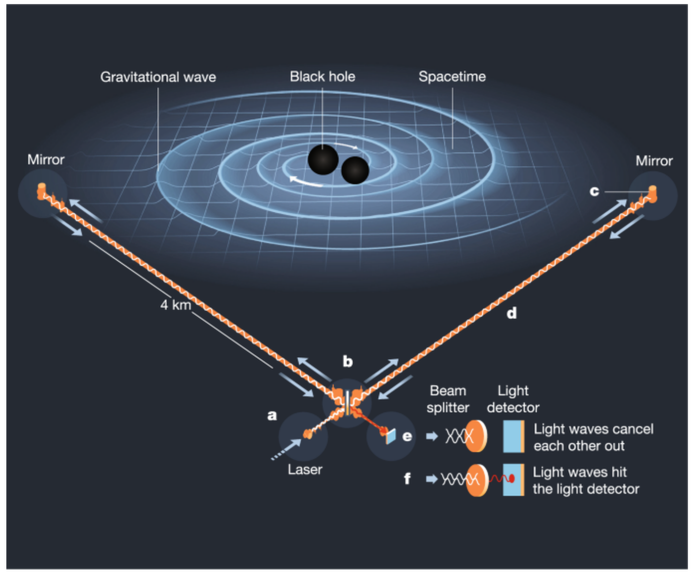
[34] P. G. Krastev, K. Gill, V. A. Villar, and E. Berger, Detection and Parameter Estimation of Gravitational Waves from Binary Neutron-Star Mergers in Real LIGO Data using Deep Learning, Phys. Lett. B **815**, 136161 (2021), arXiv:2012.13101 [astro-ph.IM].

[35] H. Shen, E. A. Huerta, E. O'Shea, P. Kumar, and Z. Zhao, Statistically-informed deep learning for gravitational wave parameter estimation, (2021), arXiv:1903.01998v3 [gr-qc].

[36] E. Cuoco, J. Powell, M. Cavaglià, K. Ackley, M. Bejger, C. Chatterjee, M. Coughlin, S. Coughlin, P. Easter, R. Essick, *et al.*, Enhancing gravitational-wave science with machine learning, Machine Learning: Science and Technology **2**, 011002 (2020), arXiv:2005.03745 [astro-ph.HE].

[36] E. Cuoco, J. Powell, M. Cavaglià, K. Ackley, M. Bejger, C. Chatterjee, M. Coughlin, S. Coughlin, P. Easter, R. Essick, *et al.*, Enhancing gravitational-wave science with machine learning, Machine Learning: Science and Technology **2**, 011002 (2020), arXiv:2005.03745 [astro-ph.HE].

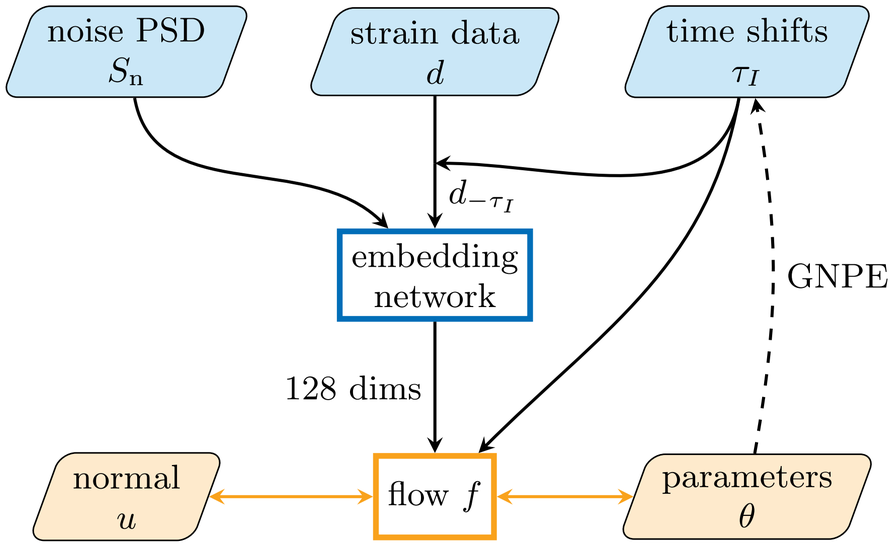
Gravitational Wave Astronomy



Real-time gravitational-wave science with neural posterior estimation

Maximilian Dax,[1,*] Stephen R. Green,[2,†] Jonathan Gair,[2,‡]
Jakob H. Macke,[1,3] Alessandra Buonanno,[2,4] and Bernhard Schölkopf[1]

[1] *Max Planck Institute for Intelligent Systems, Max-Planck-Ring 4, 72076 Tübingen, Germany*
[2] *Max Planck Institute for Gravitational Physics (Albert Einstein Institute), Am Mühlenberg 1, 14476 Potsdam, Germany*
[3] *Machine Learning in Science, University of Tübingen, 72076 Tübingen, Germany*
[4] *Department of Physics, University of Maryland, College Park, MD 20742, USA*



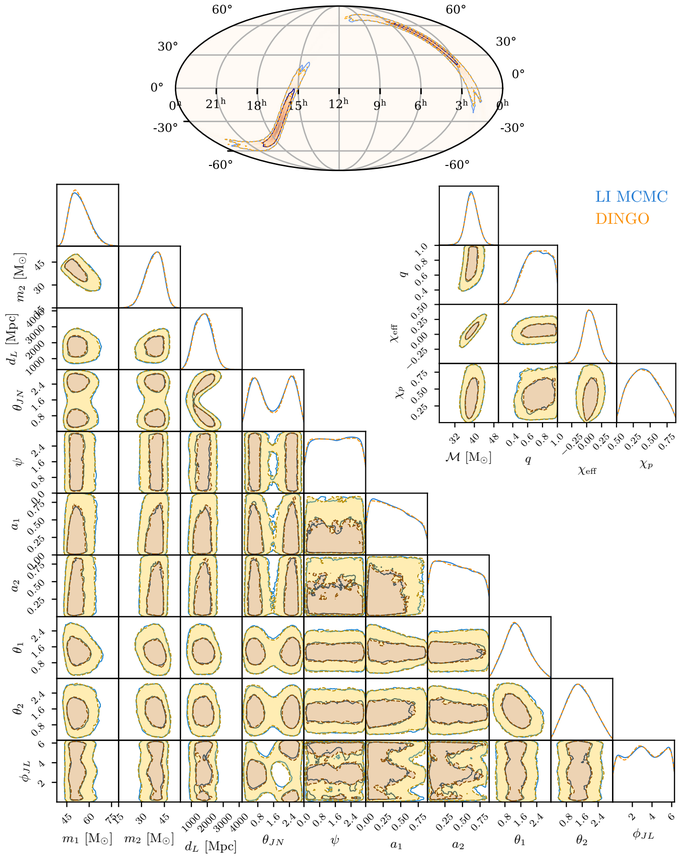


Figure 15. GW170823.

Another recent example

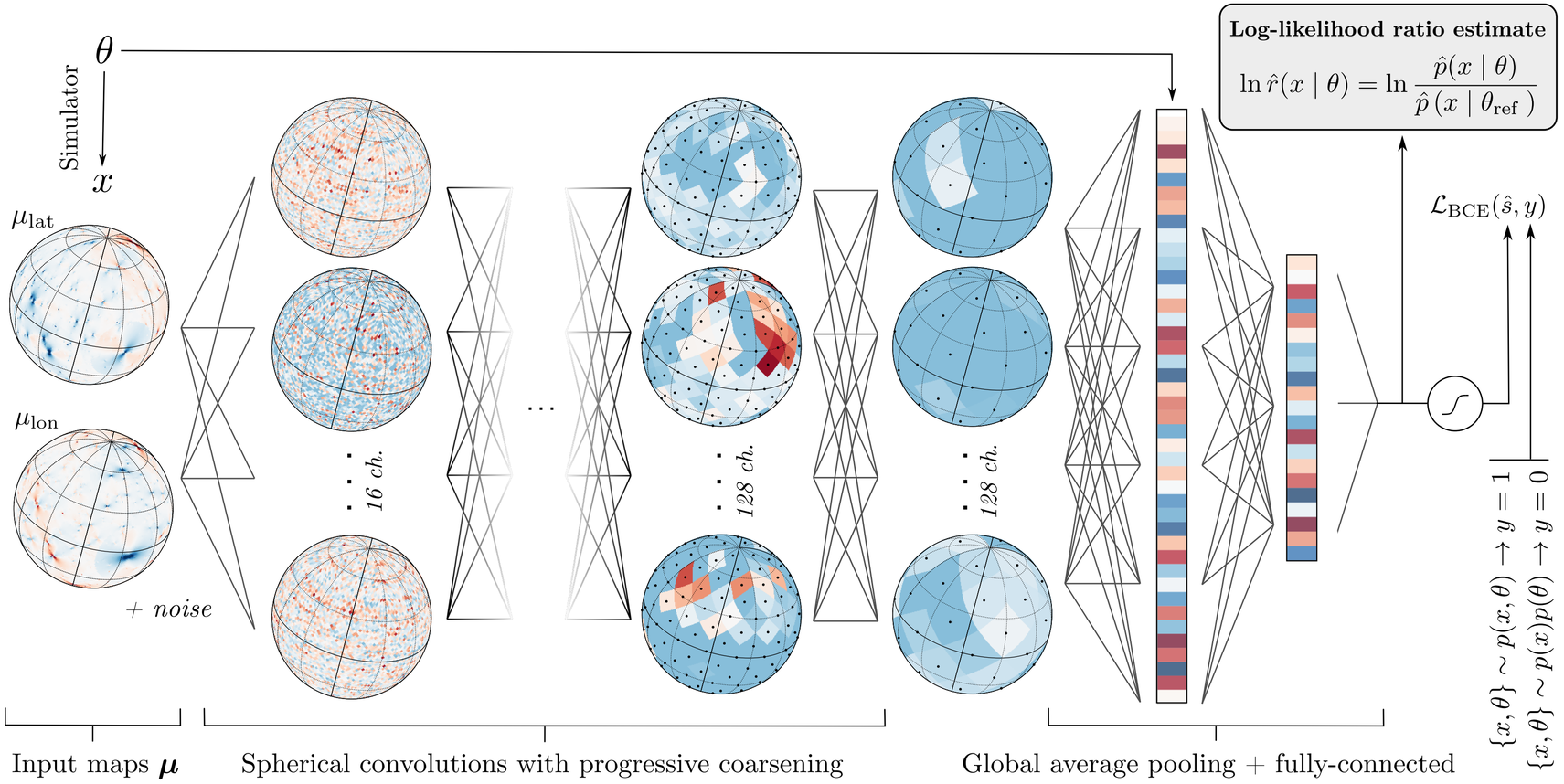
Sid Mishra-Sharma

- Neural ratio estimation
- Targets population-level parameters (fraction of dark matter in sub halos)
- Feature extractor / embedding network / learned summary statistics with inductive bias (spherical CNN)
- Aimed at future Gaia data

Inferring dark matter substructure with astrometric lensing beyond the power spectrum

Siddharth Mishra-Sharma
The NSF AI Institute for Artificial Intelligence and Fundamental Interactions
Massachusetts Institute of Technology
Harvard University
New York University
smsharma@mit.edu

[arXiv:2110.01620]



Another recent example

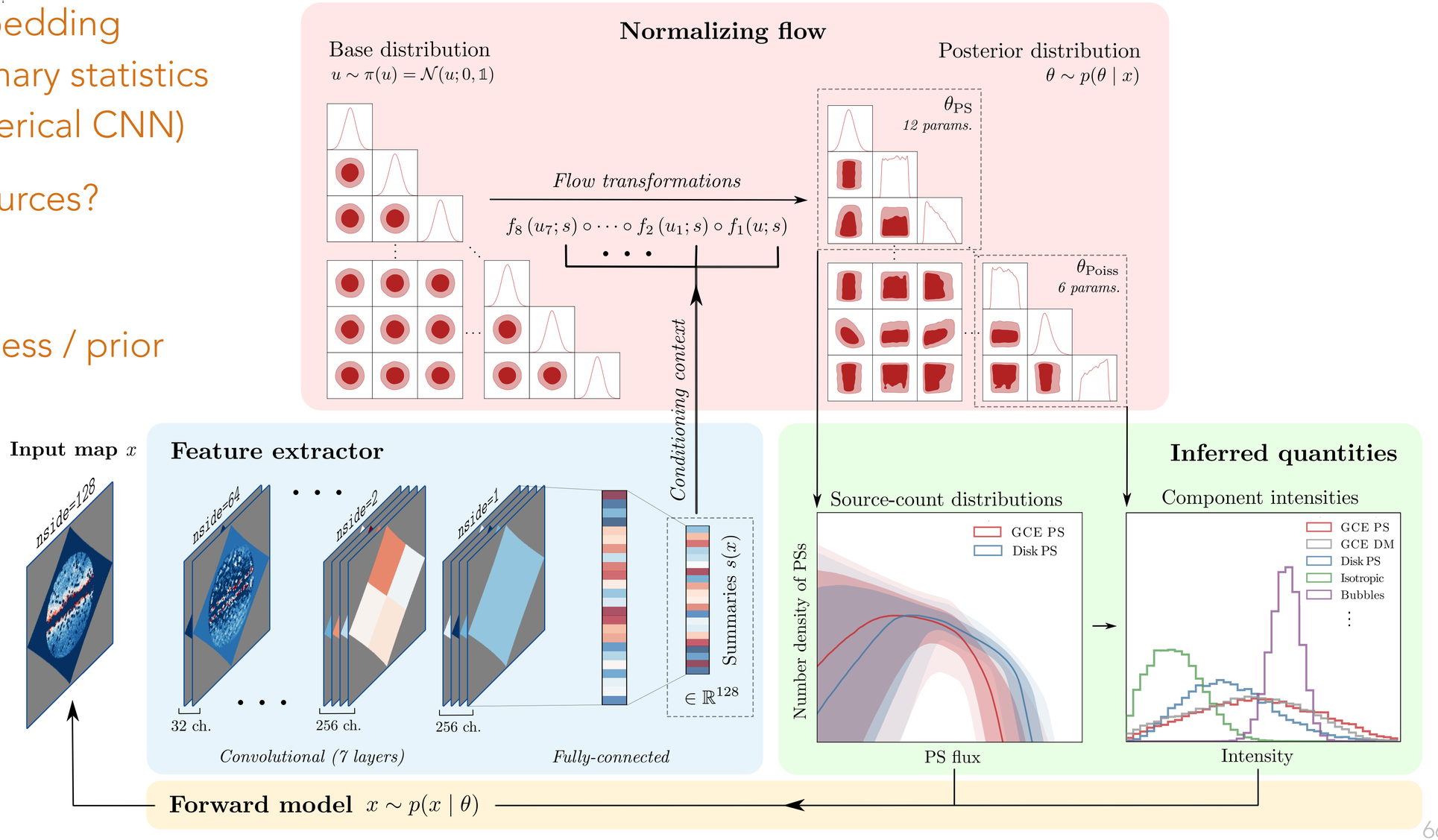
A neural simulation-based inference approach for characterizing the Galactic Center γ-ray excess

Siddharth Mishra-Sharma[1, 2, 3, 4, 5, *] and Kyle Cranmer[5, 6, †]

Sid Mishra-Sharma

[arXiv:2110.06931]

- Neural posterior estimation
- Feature extractor / embedding network / learned summary statistics with inductive bias (spherical CNN)
- Dark matter or point sources?
- Real Fermi data
- Many checks of robustness / prior sensitivity etc.



Conclusions

Simulation-based inference is a major evolution in the statistical capabilities for science enabled by advances in machine learning

- It's a great fit for HEP, multimessanger astrophysics, and computational neuroscience

New computational challenges emerge

- Want to efficiently generate simulated data and use it to train ML components in an automated way
  - Sequential approaches (active learning) can reduce simulation budget
  - Amortized inference can be very fast and has advantages in many cases
- Challenges from software environment, accelerators, workflows, etc.
- Often this workflow is embedded in active learning or decision making system

Support

United States – Israel
Binational Science Foundation

Institute for Research and Innovation
in Software for High Energy Physics

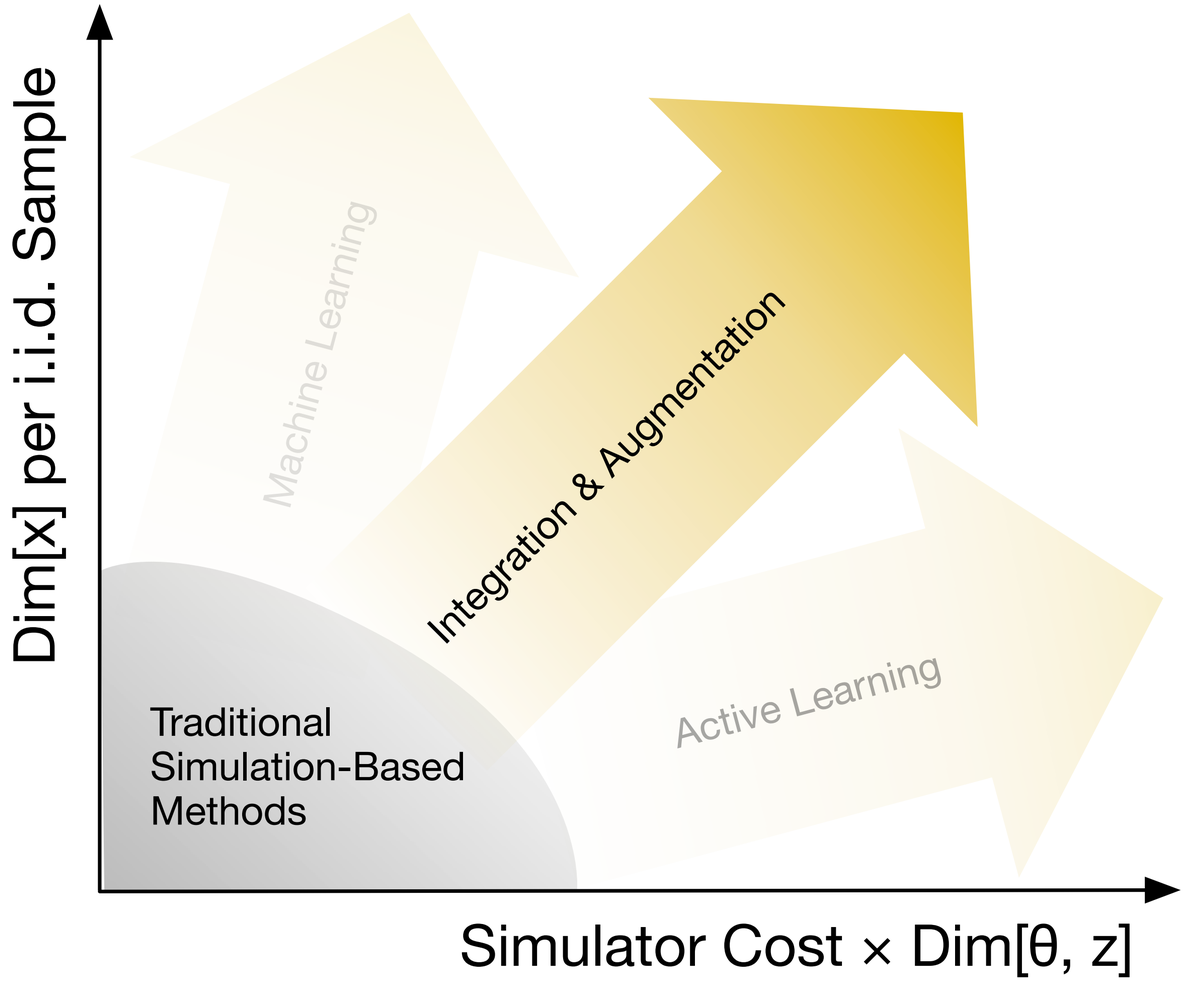
The SCAILFIN Project

scailfin.github.io

Opening the black box

Can we learn more efficiently for a fixed simulation budget?

- What if we open the black box?



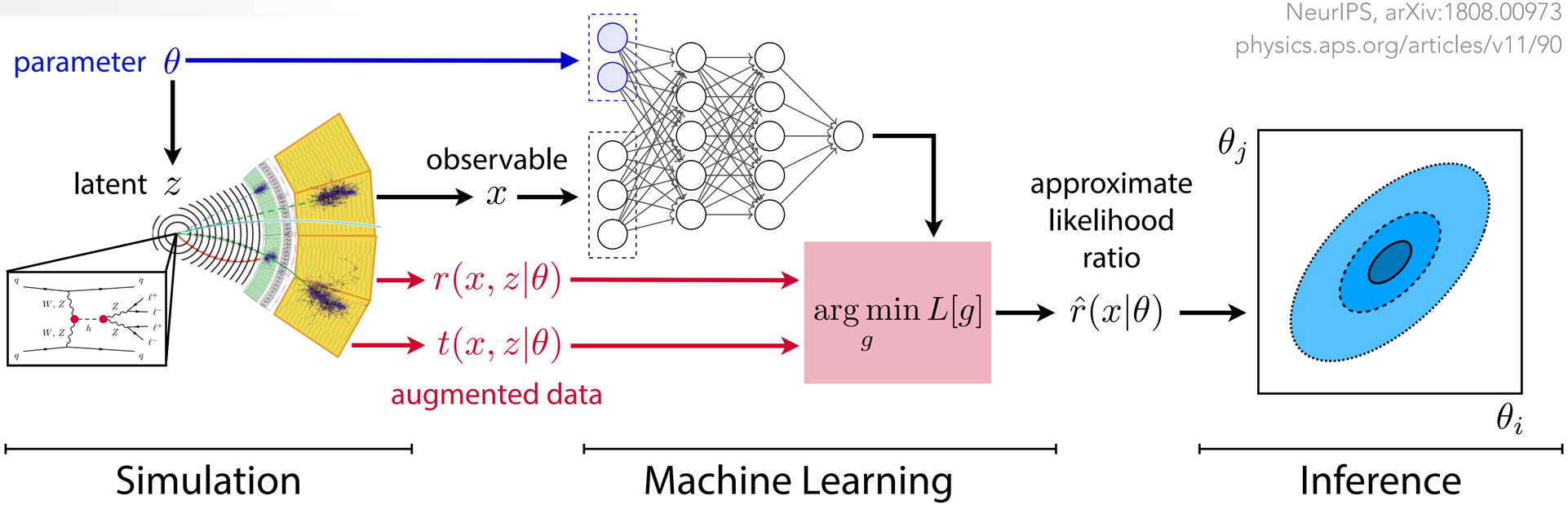
From the review

[Cranmer, J. Brehmer, G. Louppe, PNAS (2020), arXiv:1911.01429]

Fig. 3. Overview of different approaches to simulation-based inference.

Learning the likelihood ratio

PNAS, arXiv:1805.12244
PRL, arXiv:1805.00013
PRD, arXiv:1805.00020
NeurIPS, arXiv:1808.00973
physics.aps.org/articles/v11/90



Recently, we realized we can **extract more from the simulator**.

We can use **augmented data** to improve training

Johann Brehmer

Gilles Louppe

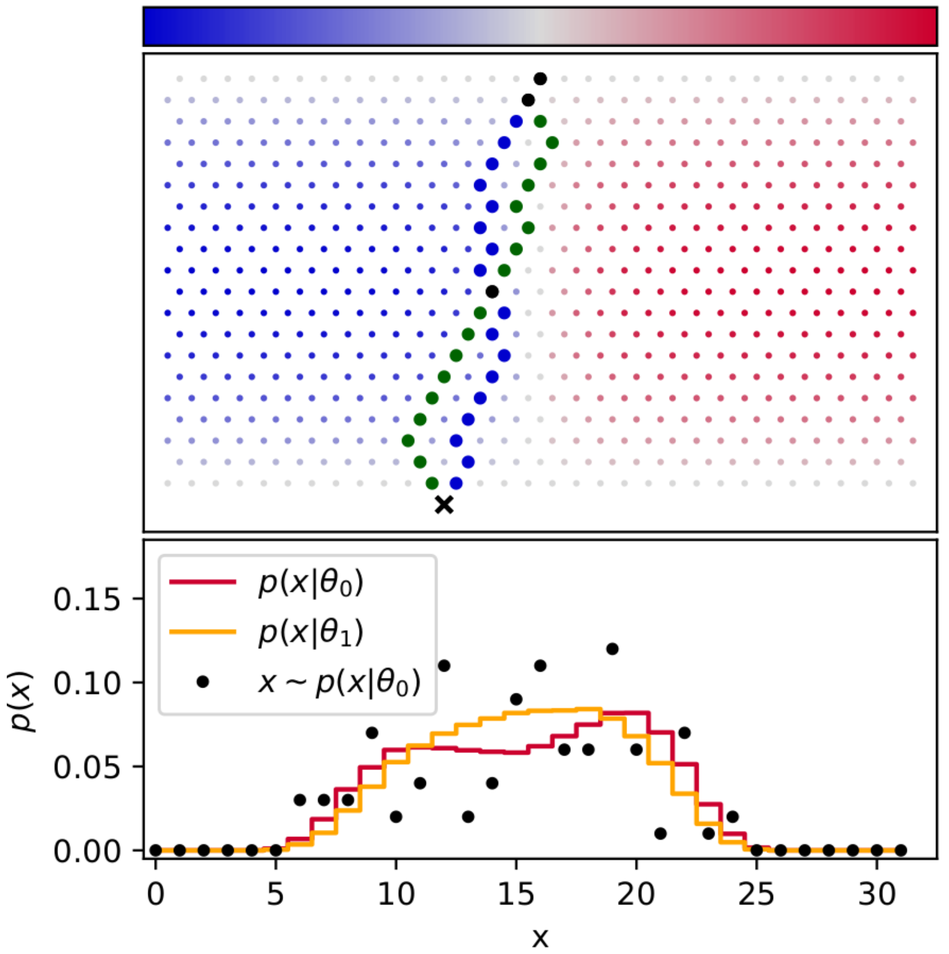
Mining Gold

Brehmer, Louppe, Pavez, KC, PNAS (2019), arXiv:1805.12244
See also Wenliang, Moskovitz, Kanagawa, Sahani, ICML2020

While implicit density is intractable

$$p(x|\theta) = \int dz p(x, z|\theta)$$

We can **augment the simulator** to calculate some quantities conditioned on latent z, which are tractable:



Joint likelihood ratio:

$$r(x, z|\theta_0, \theta_1) = \frac{p(x, z|\theta_0)}{p(x, z|\theta_1)}$$

and joint score:

$$t(x, z|\theta_0) = \frac{\nabla_\theta p(x, z|\theta)|_{\theta_0}}{p(x, z|\theta_0)} = \nabla_\theta \log p(x, z|\theta)|_{\theta_0}$$

connection to REINFORCE policy gradients

The value of gold

We can calculate the joint likelihood ratio

$$r(x, z|\theta_0, \theta_1) \equiv \frac{p(x, z_d, z_s, z_p|\theta_0)}{p(x, z_d, z_s, z_p|\theta_1)}$$

("How much more likely is this simulated event, including all intermediate states, for θ_0 compared to θ_1?")

We want the likelihood ratio function

$$r(x|\theta_0, \theta_1) \equiv \frac{p(x|\theta_0)}{p(x|\theta_1)}$$

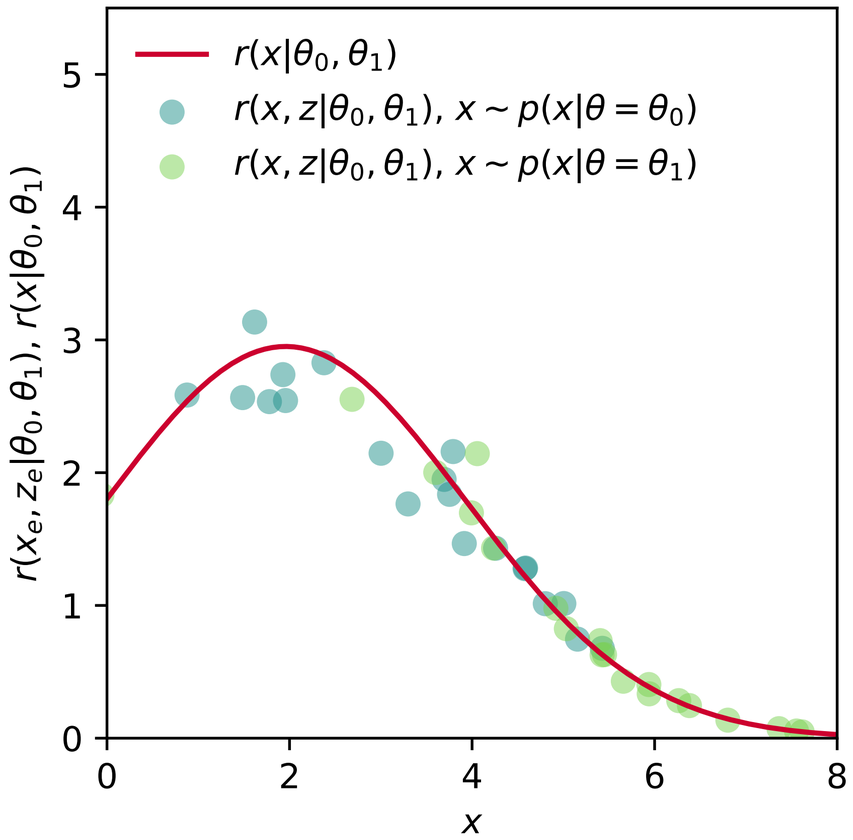
("How much more likely is the observation x for θ_0 compared to θ_1?")

The value of gold

We can calculate the joint likelihood ratio

$$r(x, z|\theta_0, \theta_1) \equiv \frac{p(x, z_d, z_s, z_p|\theta_0)}{p(x, z_d, z_s, z_p|\theta_1)}$$

$r(x, z|\theta_0, \theta_1)$ are scattered around $r(x|\theta_0, \theta_1)$



We want the likelihood ratio function

$$r(x|\theta_0, \theta_1) \equiv \frac{p(x|\theta_0)}{p(x|\theta_1)}$$

The value of gold

We can calculate the joint likelihood ratio

$$r(x, z|\theta_0, \theta_1) \equiv \frac{p(x, z_d, z_s, z_p|\theta_0)}{p(x, z_d, z_s, z_p|\theta_1)}$$

We want the likelihood ratio function

$$r(x|\theta_0, \theta_1) \equiv \frac{p(x|\theta_0)}{p(x|\theta_1)}$$

With $r(x, z|\theta_0, \theta_1)$, we define a functional like

$$L_r[\hat{r}(x|\theta_0, \theta_1)] = \int \mathrm{d}x \int \mathrm{d}z\, p(x, z|\theta_1) \left[(\hat{r}(x|\theta_0, \theta_1) - r(x, z|\theta_0, \theta_1))^2\right].$$

It is minimized by

$$r(x|\theta_0, \theta_1) = \underset{\hat{r}(x|\theta_0,\theta_1)}{\arg\min}\, L_r[\hat{r}(x|\theta_0, \theta_1)]!$$

(And we can sample from $p(x, z|\theta)$ by running the simulator.)

The value of gold

We can calculate the joint likelihood ratio

$$r(x, z|\theta_0, \theta_1) \equiv \frac{p(x, z_d, z_s, z_p|\theta_0)}{p(x, z_d, z_s, z_p|\theta_1)}$$

We want the likelihood ratio function

$$r(x|\theta_0, \theta_1) \equiv \frac{p(x|\theta_0)}{p(x|\theta_1)}$$

With $r(x, z|\theta_0, \theta_1)$, we define a functional like

$$L_r[\hat{r}(x|\theta_0, \theta_1)] = \int \mathrm{d}x \int \mathrm{d}z\, p(x, z|\theta_1) \left[(\hat{r}(x|\theta_0, \theta_1) - r(x, z|\theta_0, \theta_1))^2\right].$$

It is minimized by

$$r(x|\theta_0, \theta_1) = \underset{\hat{r}(x|\theta_0,\theta_1)}{\arg\min}\, L_r[\hat{r}(x|\theta_0, \theta_1)]!$$

(And we can sample from $p(x, z|\theta)$ by running the simulator.)

.... and then magic ...

$$\begin{aligned}\mathbb{E}_{z\sim p(z|x,\theta_1)}\left[r(x, z|\theta_0, \theta_1)\right] &= \int \mathrm{d}z\, p(z|x, \theta_1)\, \frac{p(x, z|\theta_0)}{p(x, z|\theta_1)} \\ &= \int \mathrm{d}z\, \frac{p(x, z|\theta_1)}{p(x|\theta_1)}\, \frac{p(x, z|\theta_0)}{p(x, z|\theta_1)} \\ &= r(x|\theta_0, \theta_1)\ \textbf{!}\end{aligned}$$

Learning the score

Similar to the joint likelihood ratio, from the simulator we can extract the joint score

$$t(x, z|\theta_0) \equiv \nabla_\theta \log p(x, z_d, z_s, z_p|\theta)\Big|_{\theta_0}$$

We want the score

$$t(x|\theta_0) \equiv \nabla_\theta \log p(x|\theta)\Big|_{\theta_0}$$

Learning the score

Similar to the joint likelihood ratio, from the simulator we can extract the joint score

$$t(x, z|\theta_0) \equiv \nabla_\theta \log p(x, z_d, z_s, z_p|\theta)\Big|_{\theta_0}$$

We want the score

$$t(x|\theta_0) \equiv \nabla_\theta \log p(x|\theta)\Big|_{\theta_0}$$

Given $t(x, z|\theta_0)$,
we define the functional

$$L_t[\hat{t}(x|\theta_0)] = \int \mathrm{d}x \int \mathrm{d}z \;\; p(x, z|\theta_0) \;\; \Big[\big(\hat{t}(x|\theta_0) - t(x, z|\theta_0)\big)^2\Big].$$

One can show it is minimized by

$$t(x|\theta_0) = \arg\min_{\hat{t}(x|\theta_0)} L_t[\hat{t}(x|\theta_0)]\,.$$

Again, we implement this minimization
through machine learning.

Augmented Training Data

Brehmer, Louppe, Pavez, KC, PNAS (2019), arXiv:1805.12244