

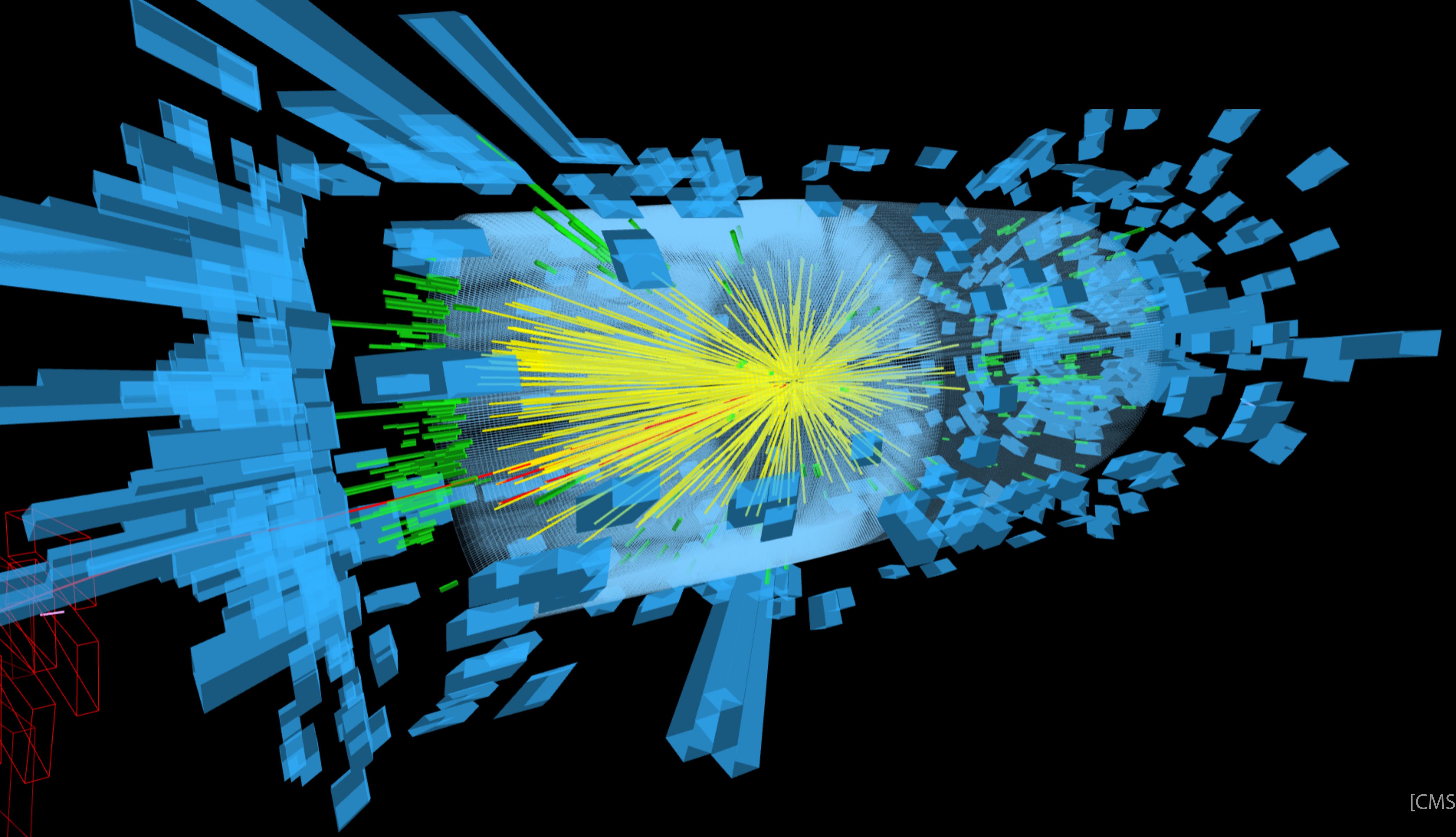
How machine learning can help us get the most out of high-precision physics models

Johann Brehmer

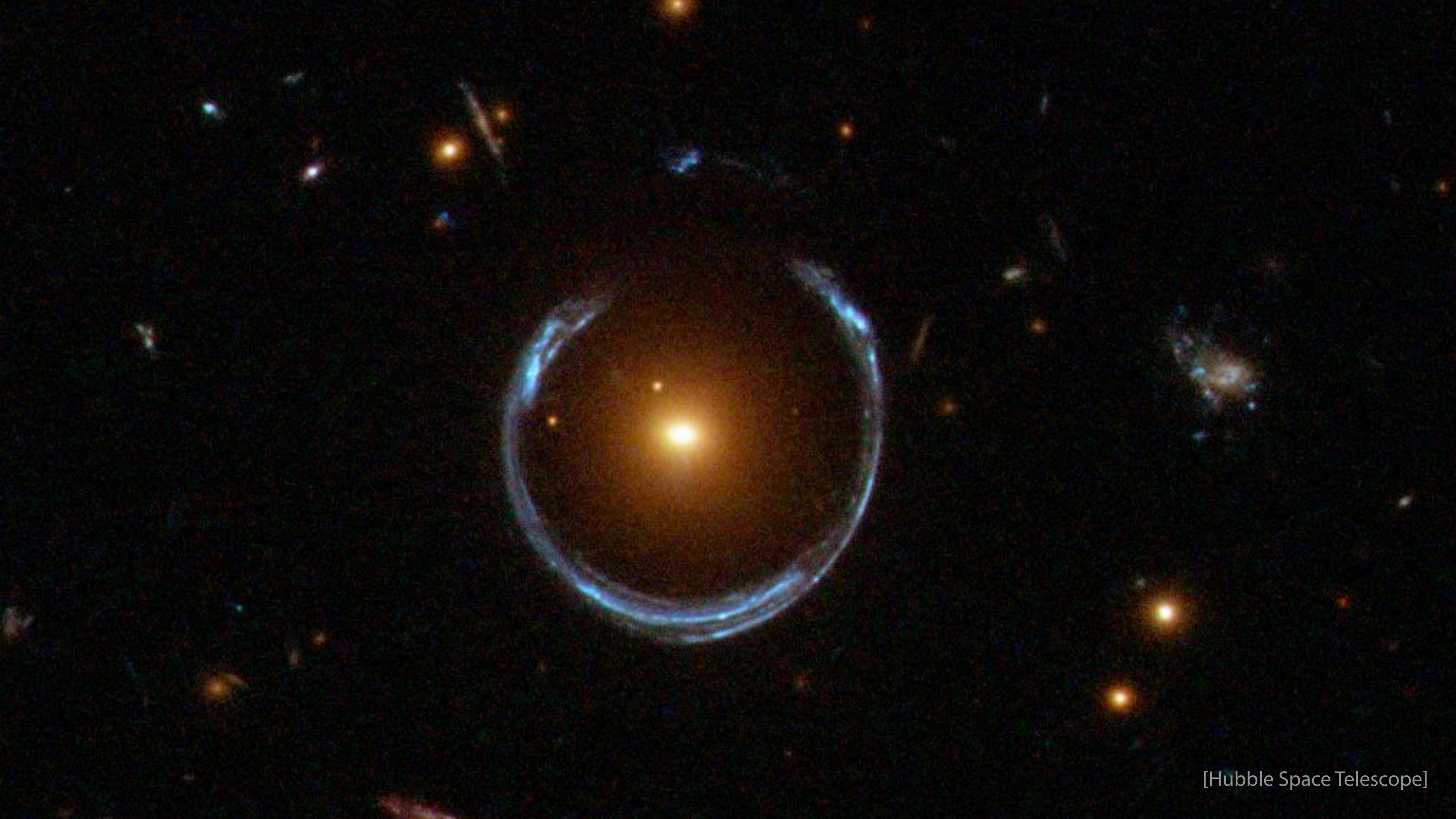
New York University

Qualcomm AI Research

September 2, 2020

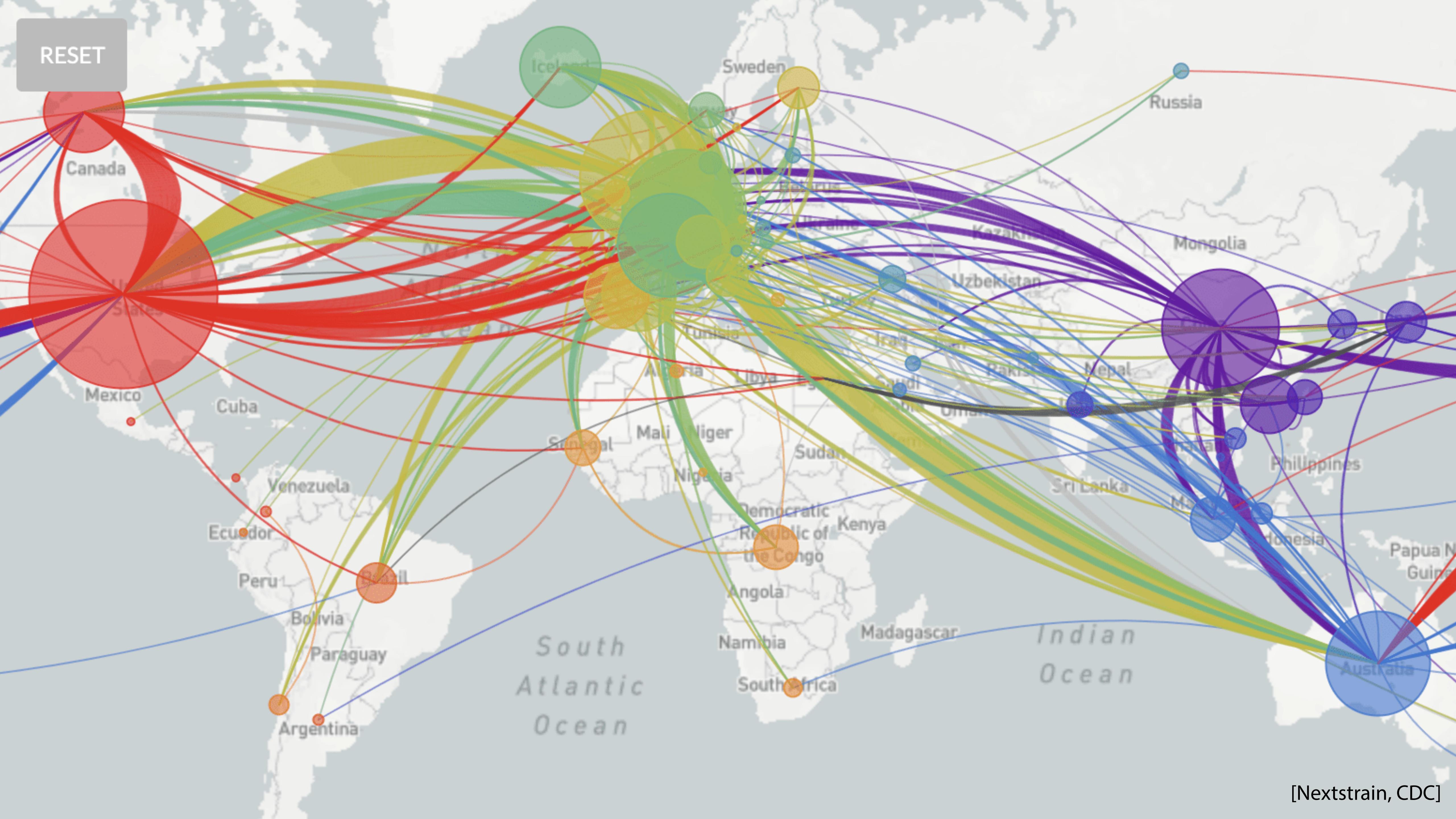


[CMS]



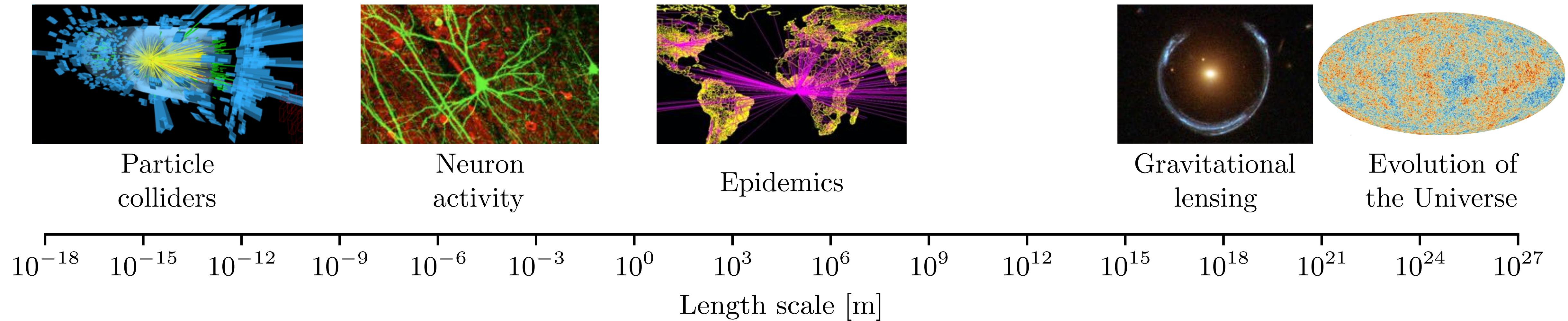
[Hubble Space Telescope]

RESET



Big picture

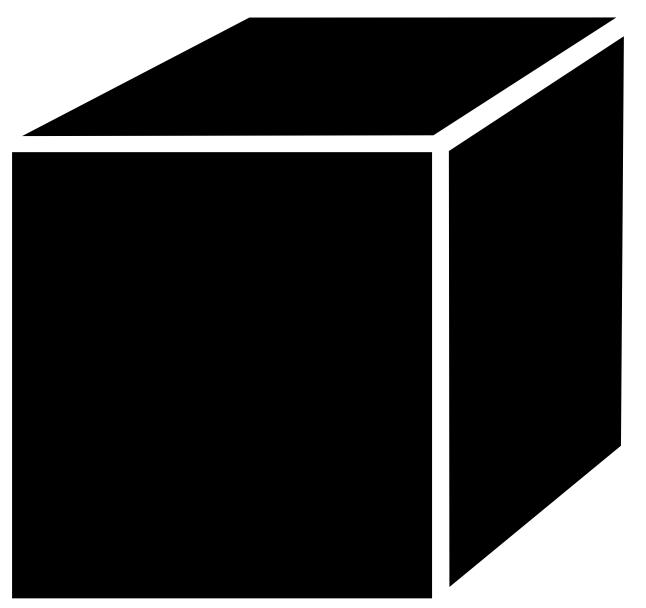
[K. Cranmer, JB, G. Louppe 1911.01429]



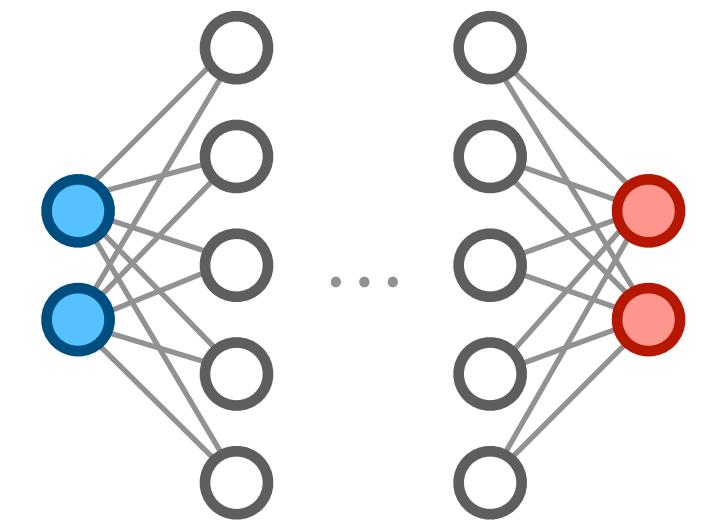
Simulators give high-precision predictions for many phenomena in science and engineering.

Unfortunately, they are poorly suited for inference.

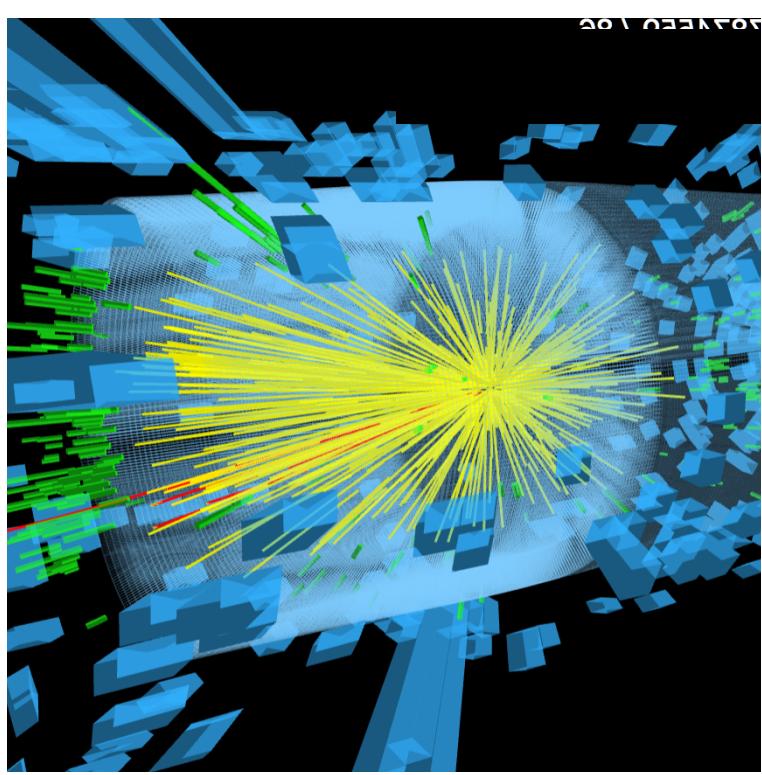
Machine learning can provide efficient models for powerful inference algorithms.



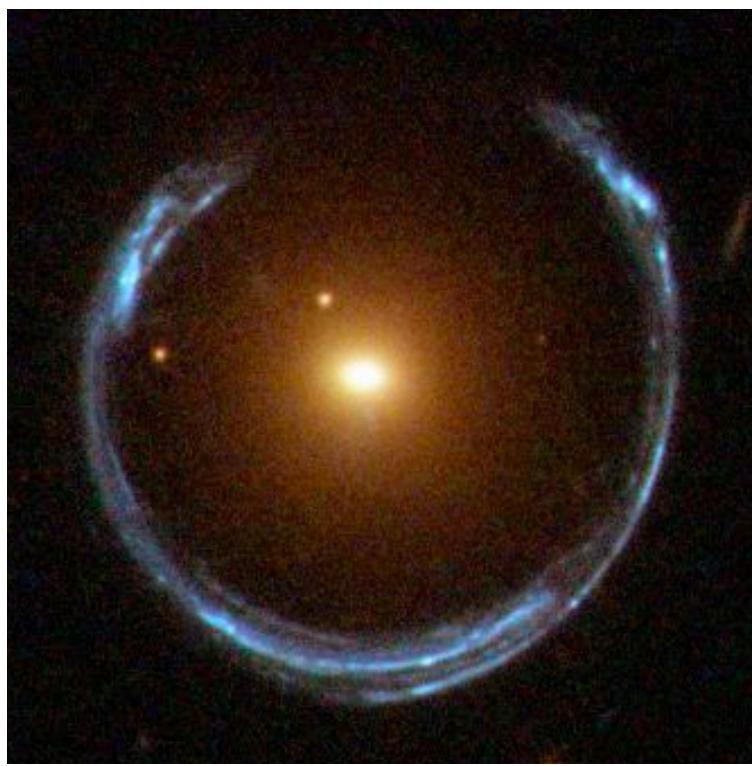
1. The simulation-based inference problem



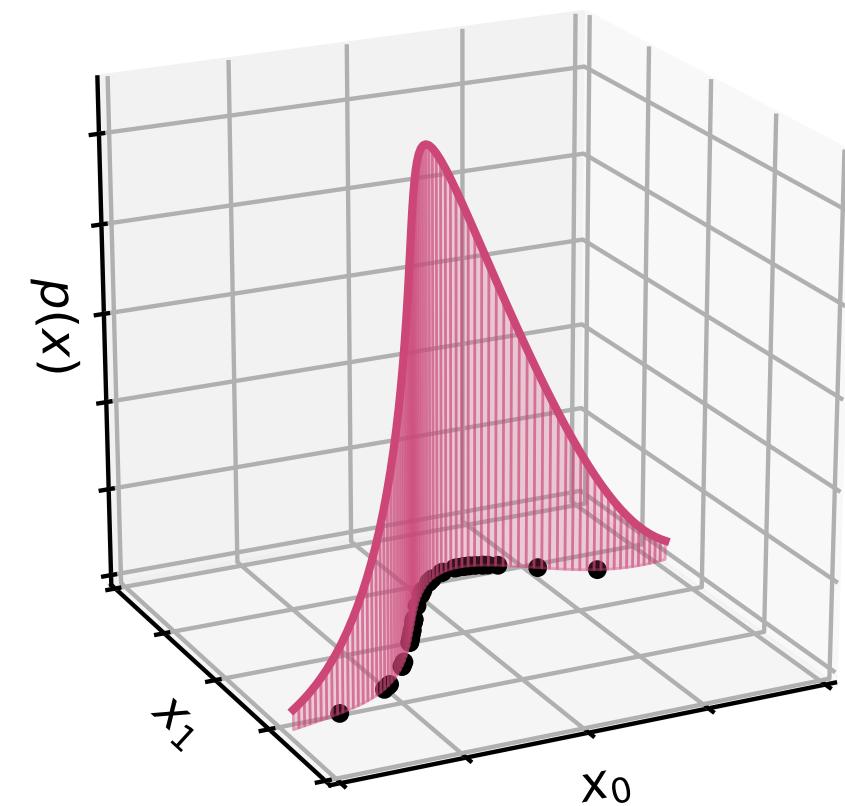
2. Machine learning methods



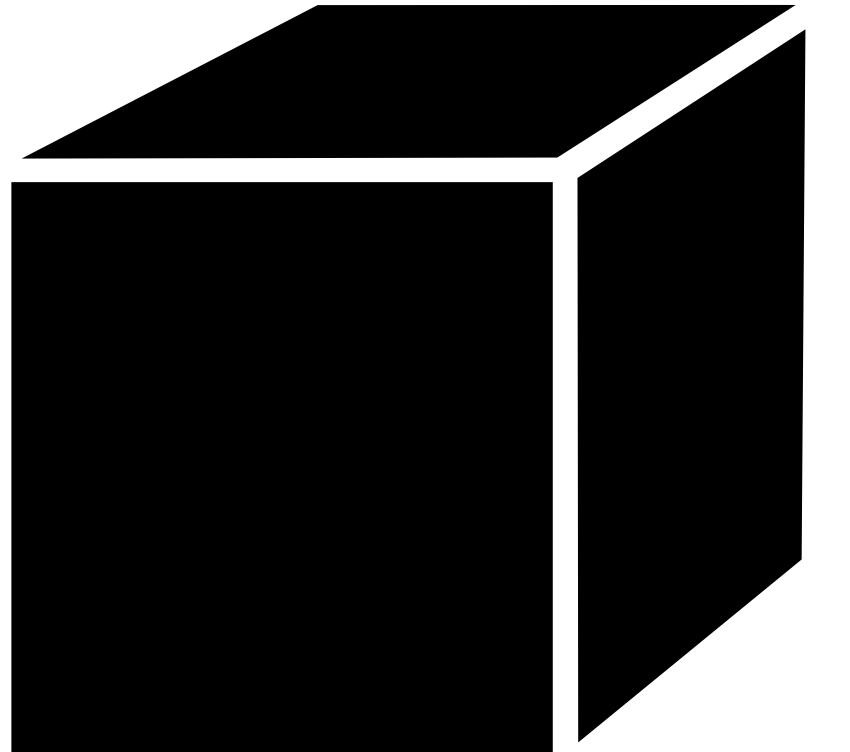
3. Particle physics



4. Astrophysics

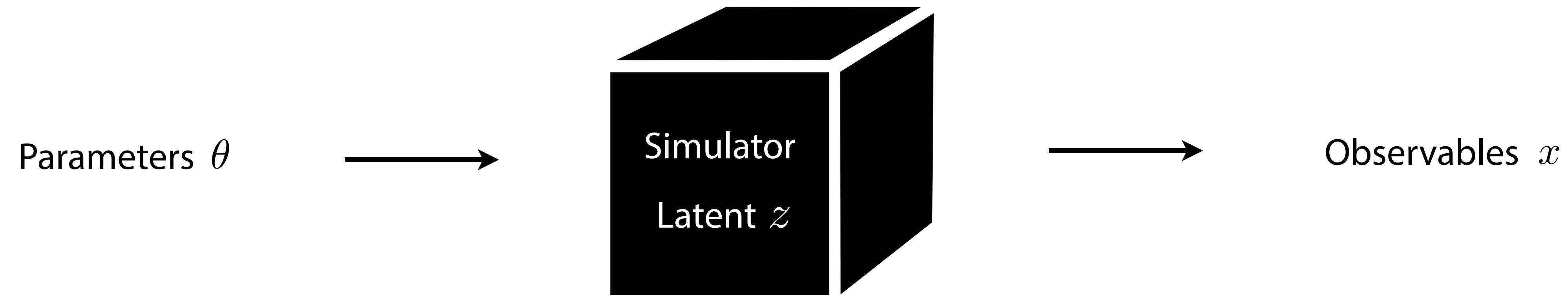


5. Tangents

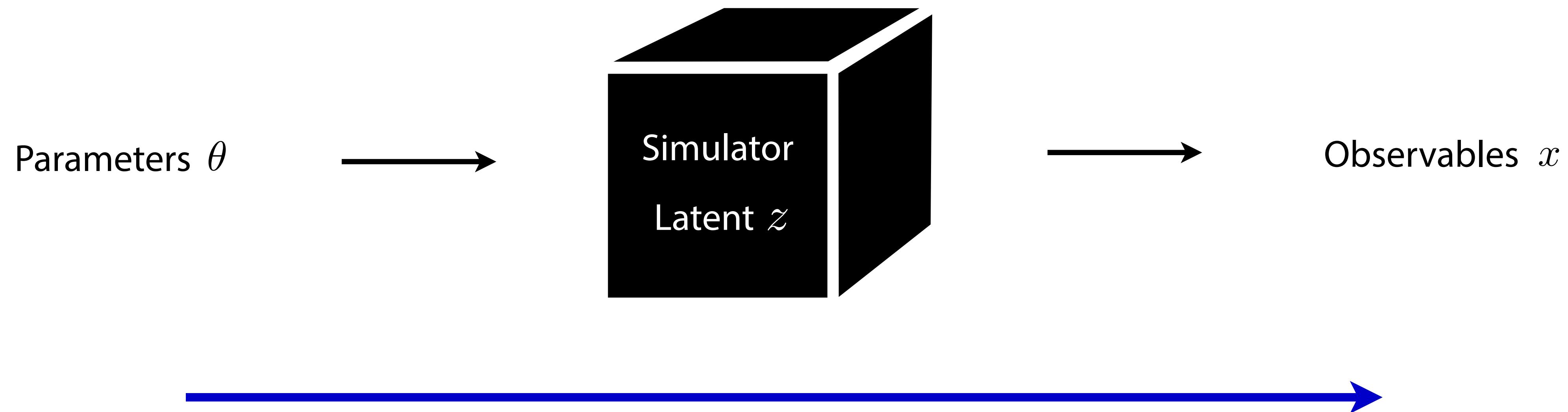


1. The simulation-based inference problem

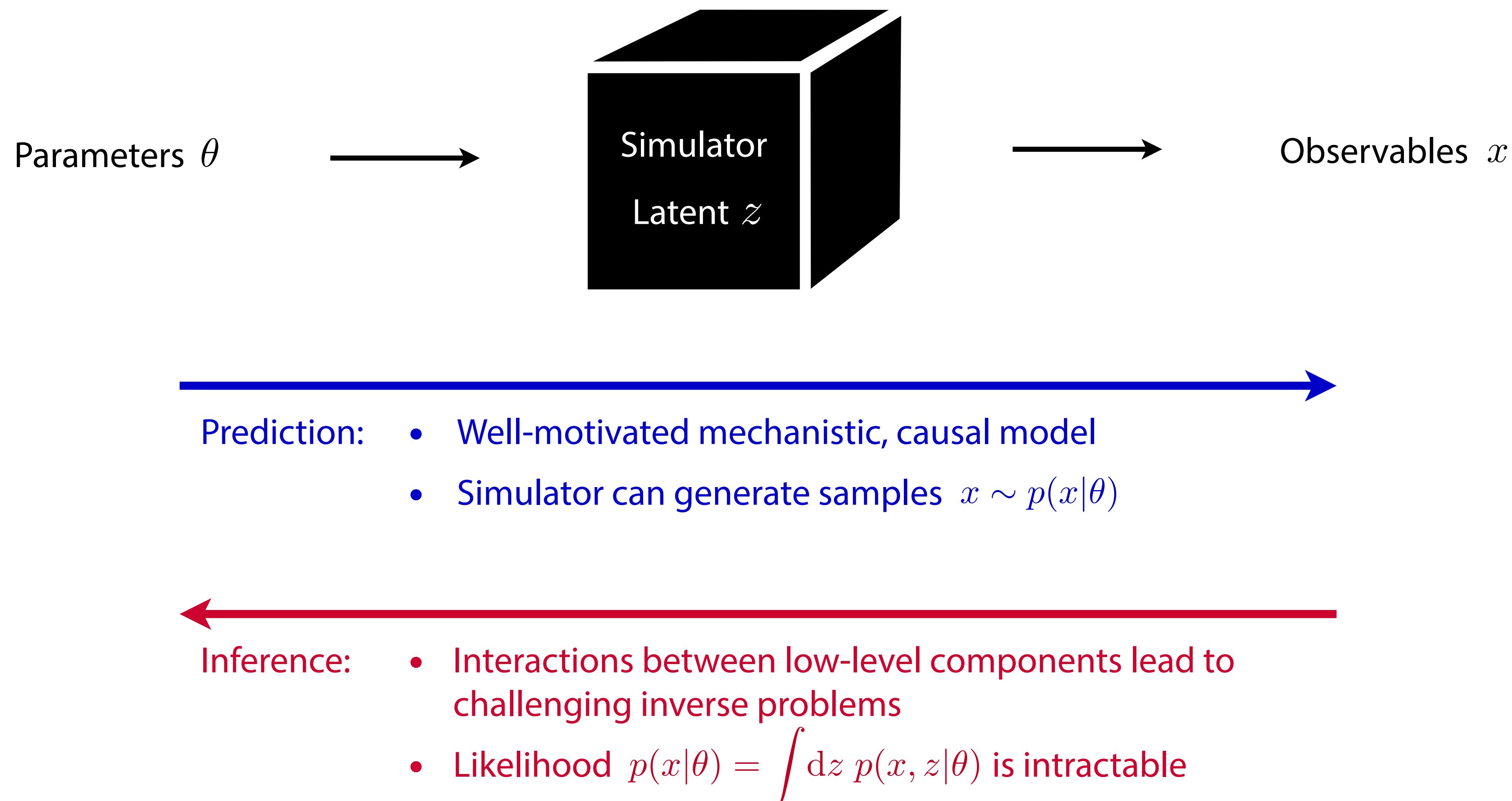
The problem of simulation-based (“likelihood-free”) inference



The problem of simulation-based (“likelihood-free”) inference



The problem of simulation-based (“likelihood-free”) inference



Three problem statements

Given

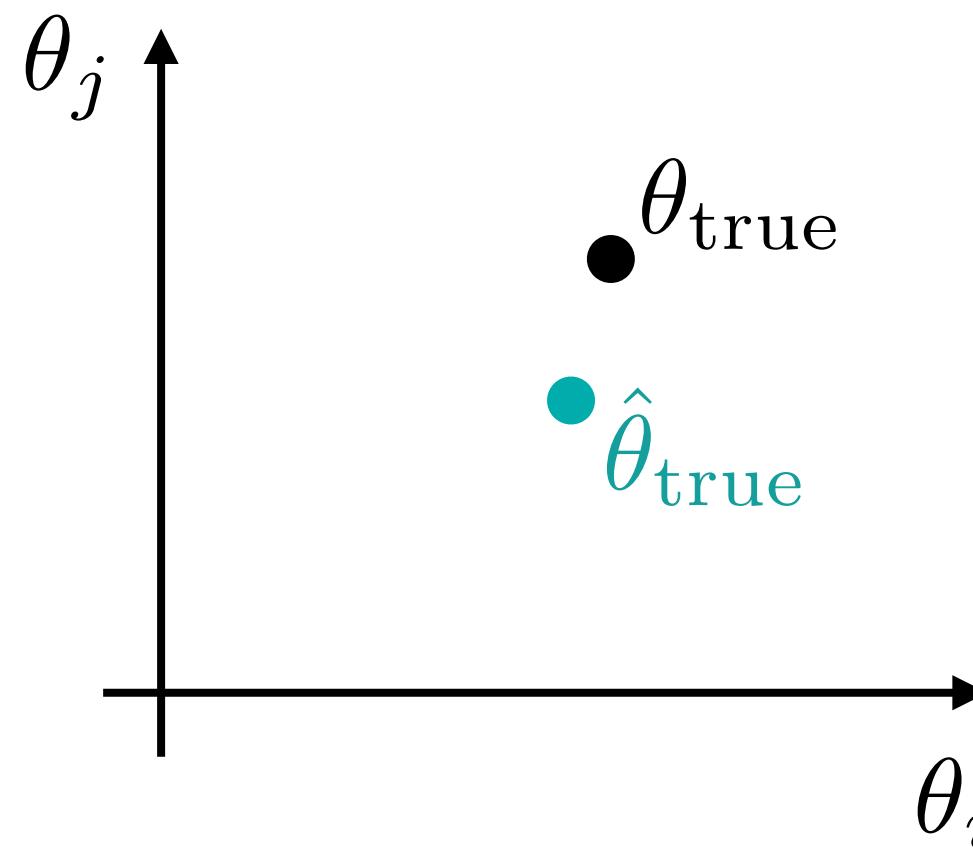
- a simulator that lets you generate N samples $x_i \sim p(x_i|\theta_i)$ (for parameters θ_i of our choice),
- observed data $x_{\text{obs}} \sim p(x_{\text{obs}}|\theta_{\text{true}})$, and
- a prior $p(\theta)$,

Three problem statements

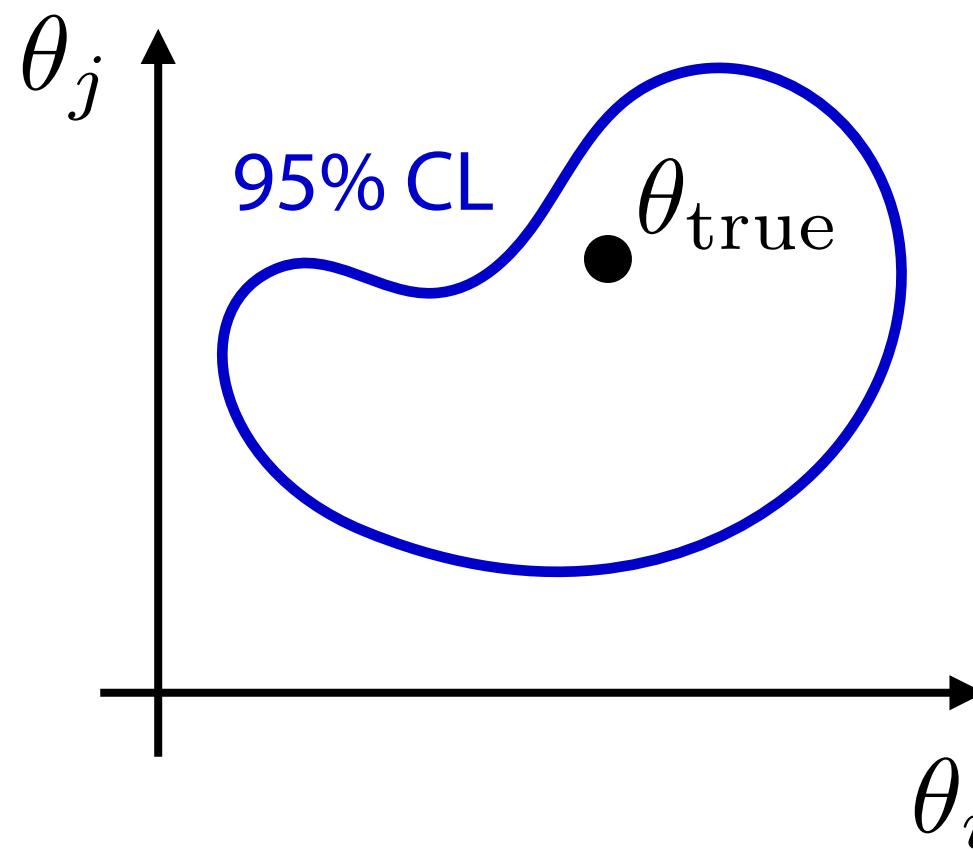
Given

- a simulator that lets you generate N samples $x_i \sim p(x_i|\theta_i)$ (for parameters θ_i of our choice),
- observed data $x_{\text{obs}} \sim p(x_{\text{obs}}|\theta_{\text{true}})$, and
- a prior $p(\theta)$,

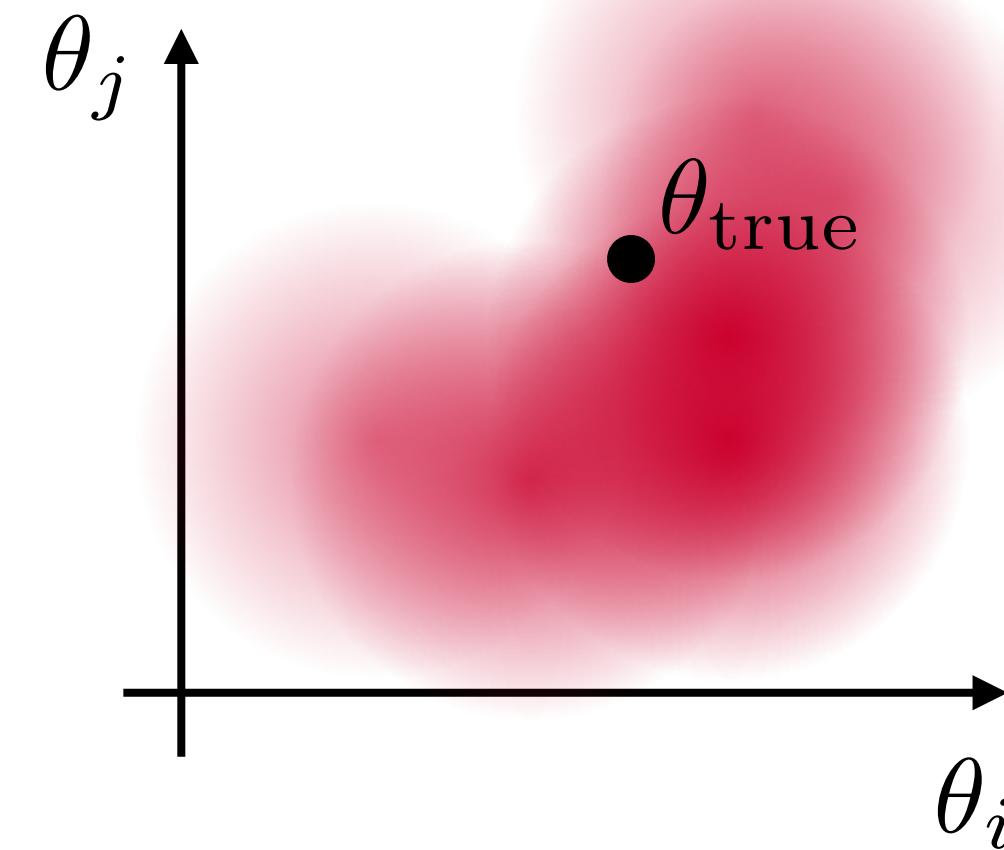
a) estimate $\hat{\theta}_{\text{true}}$
(e.g. MLE)

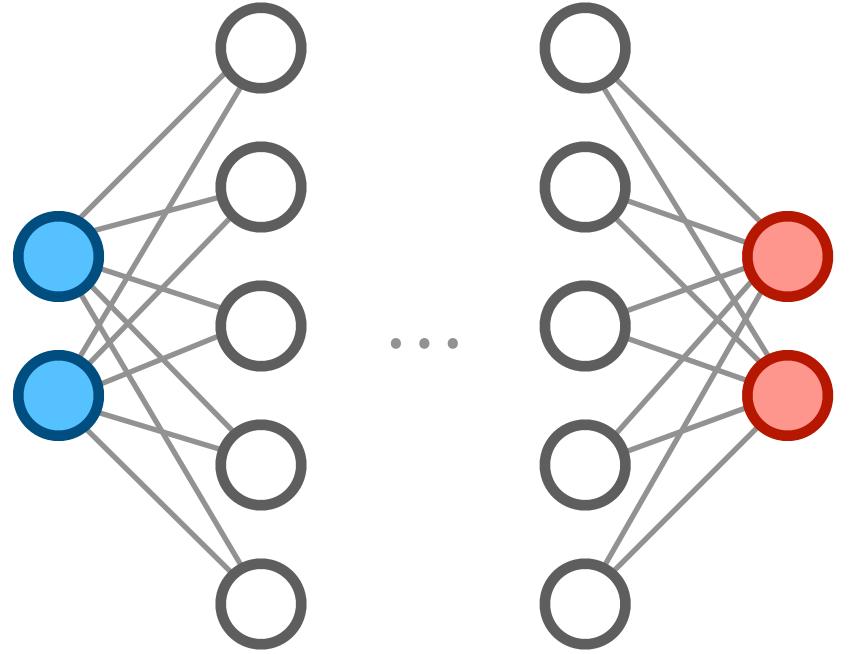


b) construct confidence sets
(e.g. likelihood ratio tests)



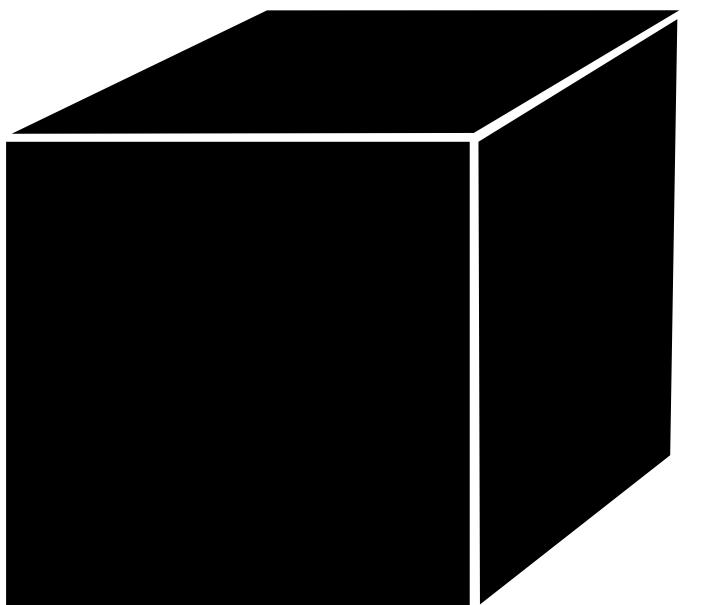
c) estimate the posterior
(or sample from posterior)





2. Machine learning methods

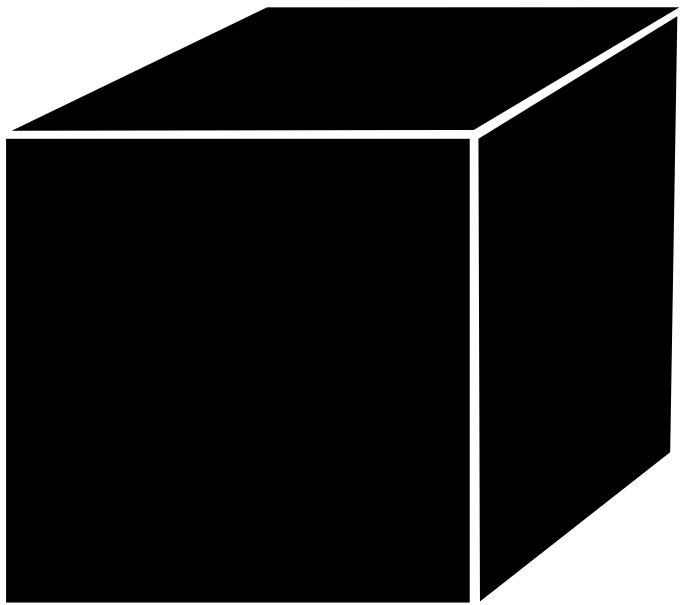
Get the best of two worlds



Simulators: focus on understanding

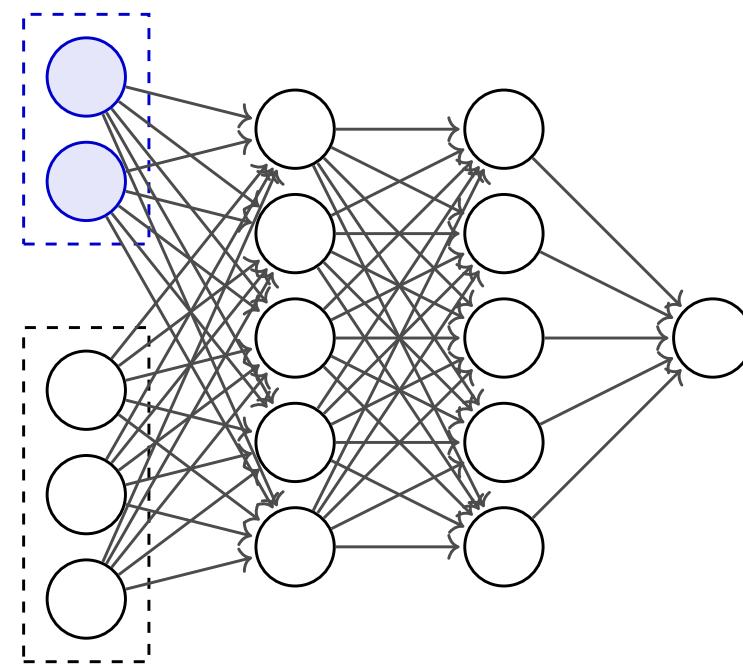
- based on mechanistic, causal model
- interpretable parameters

Get the best of two worlds



Simulators: focus on understanding

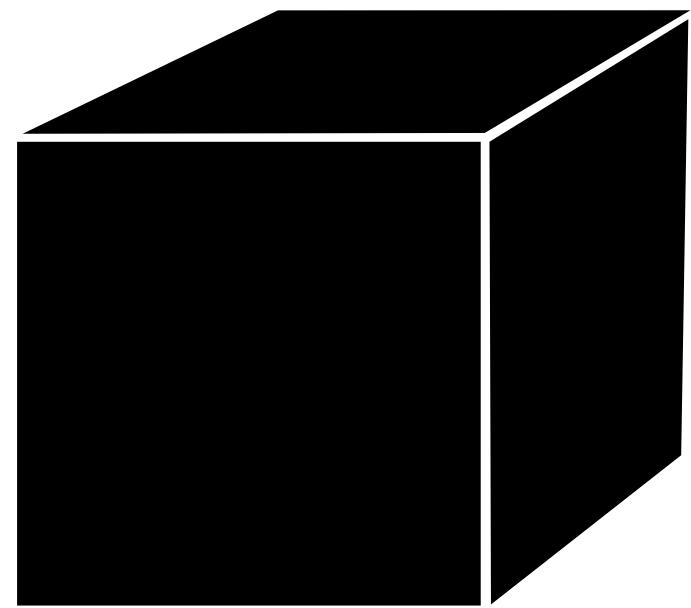
- based on mechanistic, causal model
- interpretable parameters



Machine learning models: focus on performance

- good at learning representations from data
- good inductive biases (images, sequences, graphs, symmetries, hierarchical structures...)
- differentiable, often invertible, probabilistic: well-suited for inference / fitting

Get the best of two worlds

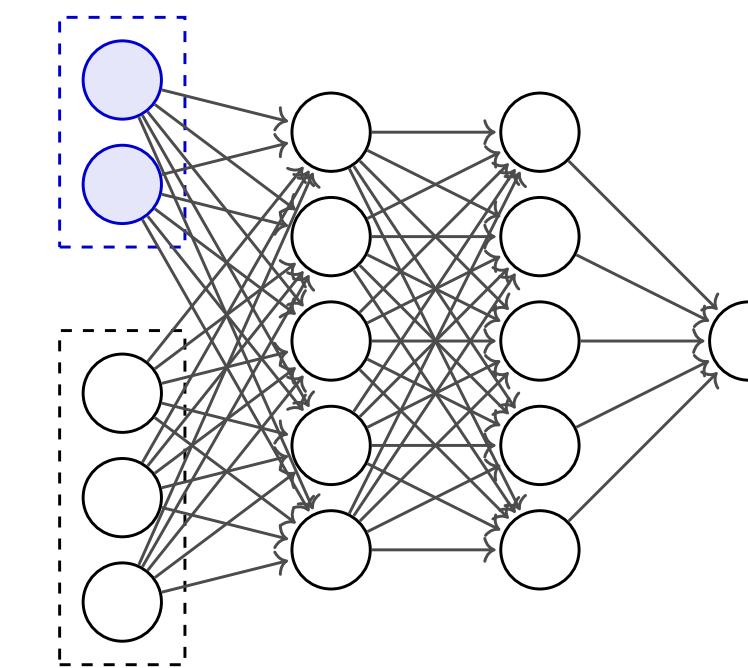


Can we use ML
models to fit
simulators to data?

Simulators: focus on understanding

- based on mechanistic, causal model
- interpretable parameters

Can we inject
domain knowledge
into ML models?

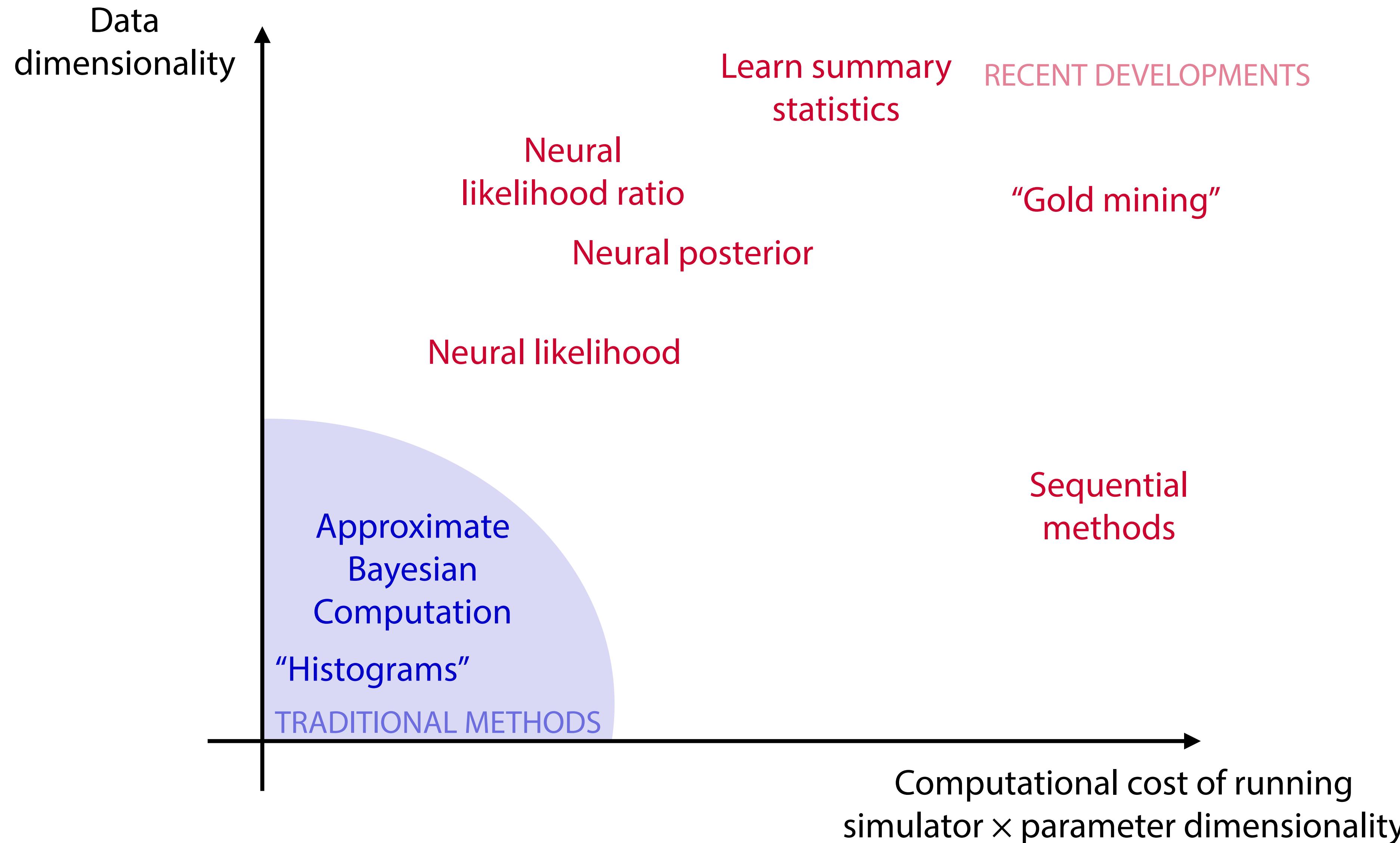


Machine learning models: focus on performance

- good at learning representations from data
- good inductive biases (images, sequences, graphs, symmetries, hierarchical structures...)
- differentiable, often invertible, probabilistic: well-suited for inference / fitting

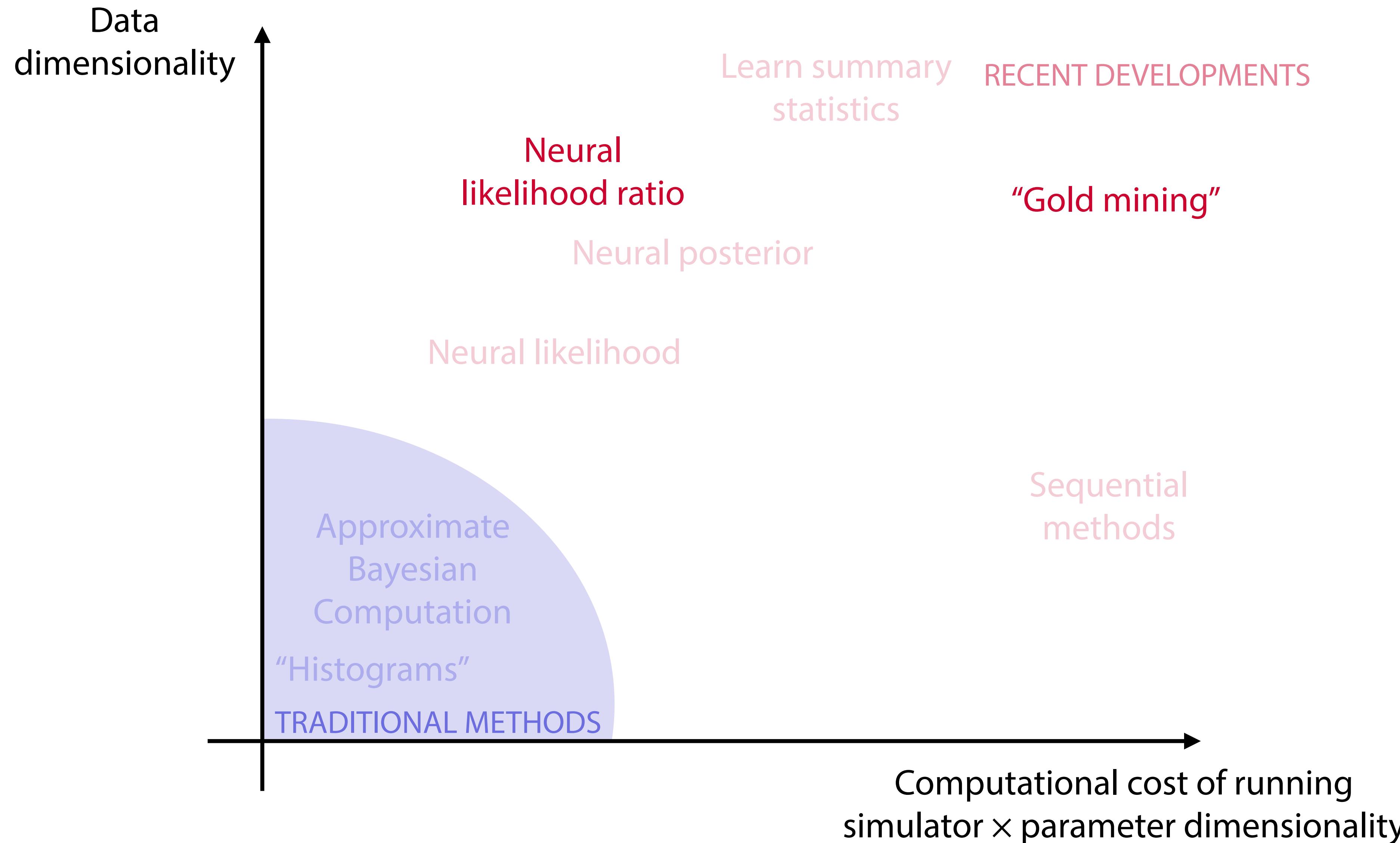
Simulation-based inference methods

[K. Cranmer, JB, G. Louppe 1911.01429]



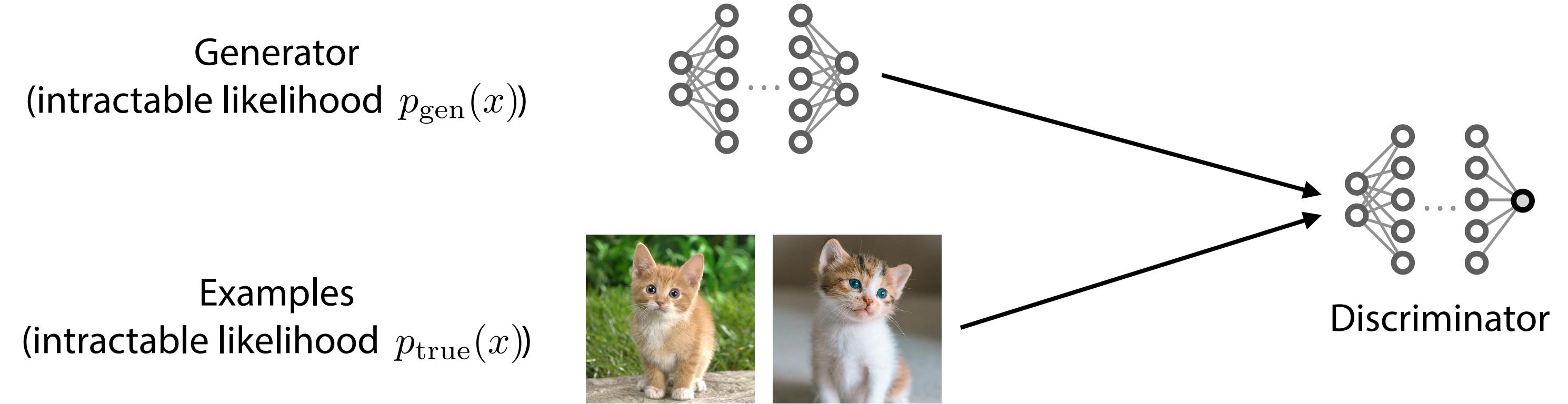
Simulation-based inference methods

[K. Cranmer, JB, G. Louppe 1911.01429]



Idea 1: the likelihood ratio trick

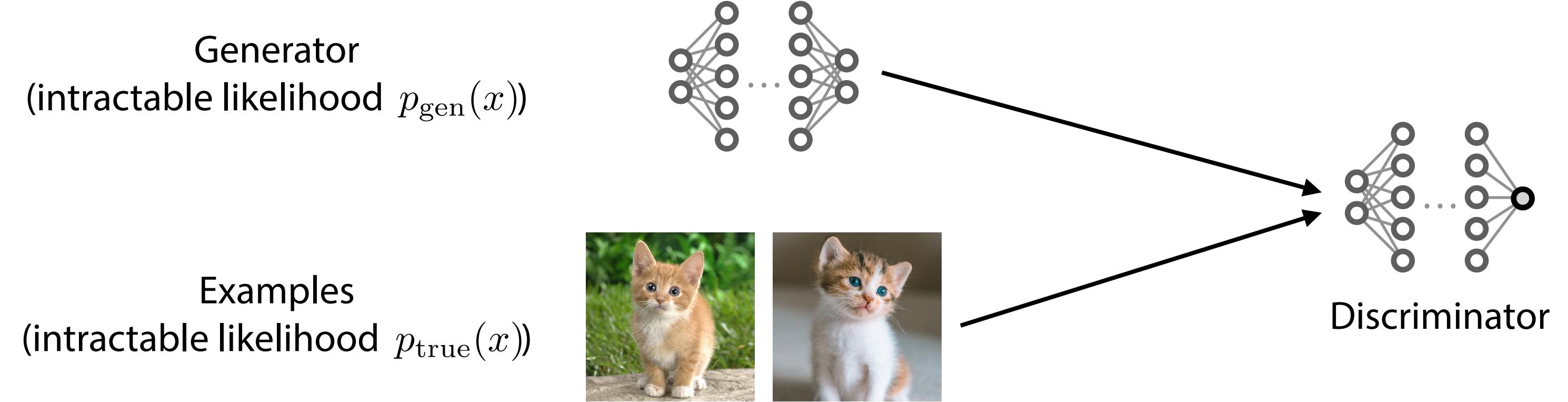
- Generative Adversarial Networks (GANs):



[I. Goodfellow et al. 1406.2661]

Idea 1: the likelihood ratio trick

- Generative Adversarial Networks (GANs):



[I. Goodfellow et al. 1406.2661]

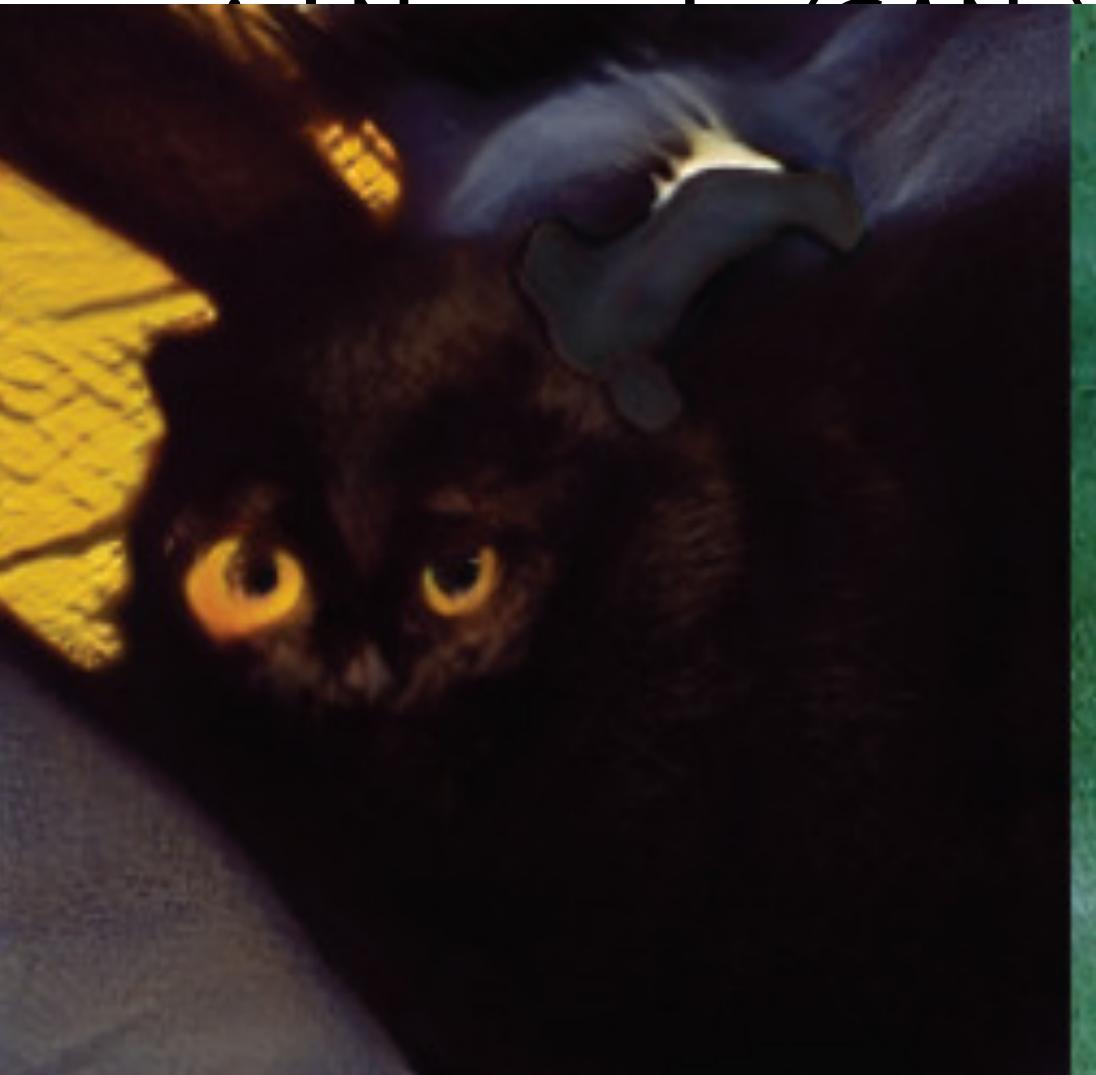
Discriminator learns decision function

$$s(x) \rightarrow \frac{p_{\text{true}}(x)}{p_{\text{gen}}(x) + p_{\text{true}}(x)}$$

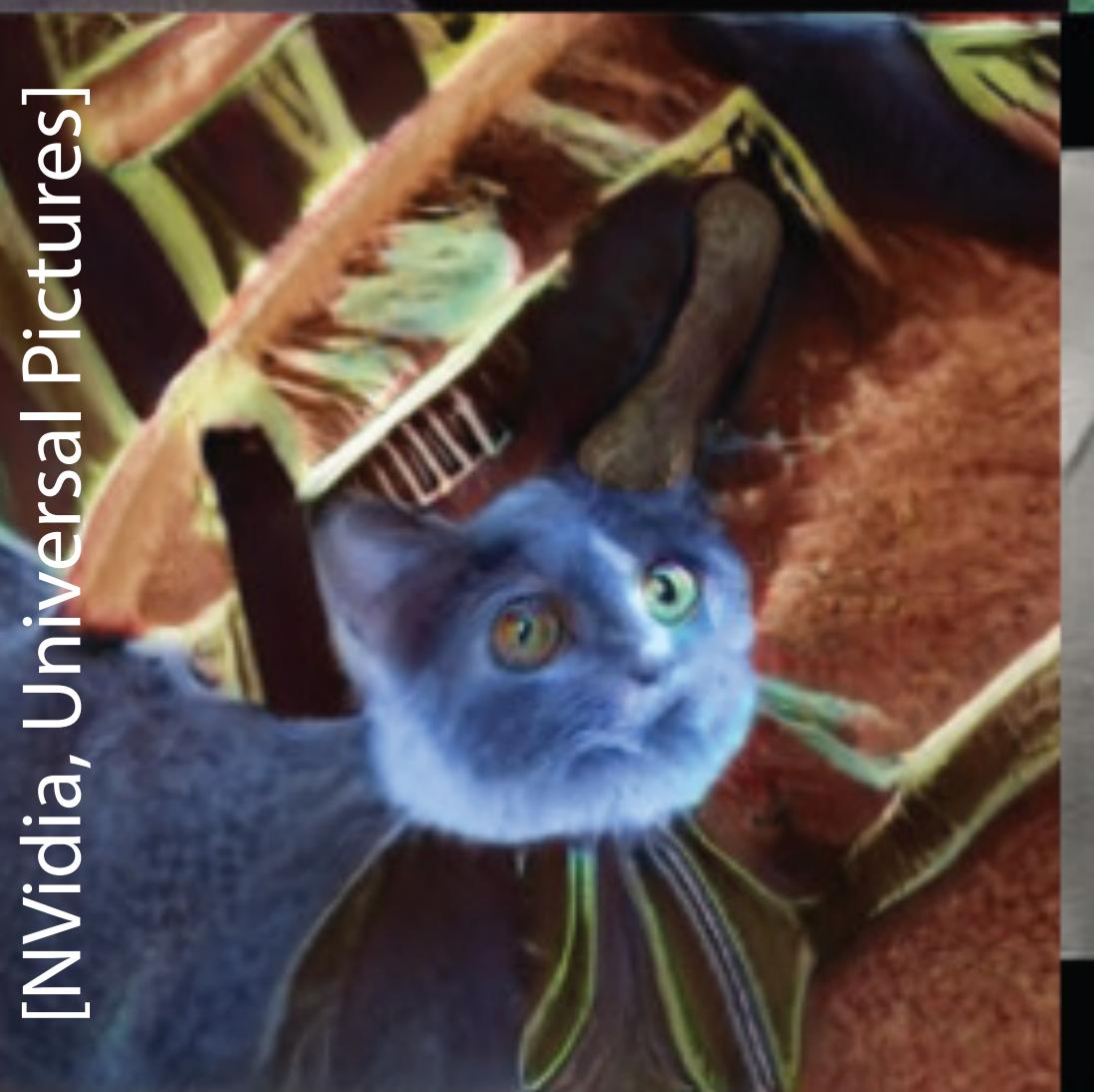
Idea 1: the likelihood ratio trick

- Generative Adversarial Networks (GANs)

Generator
(intractable likelihood $p_g(x)$)

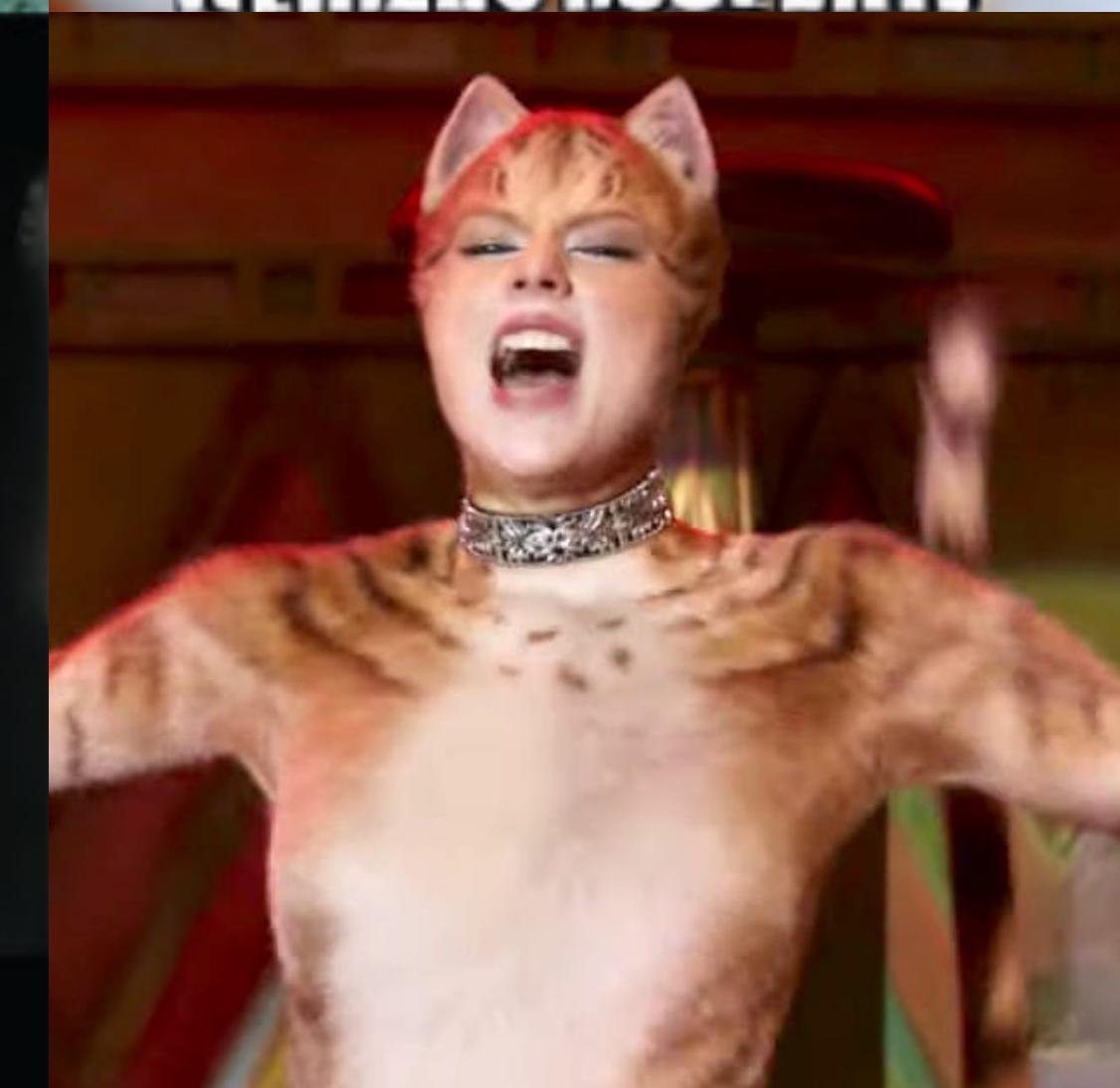
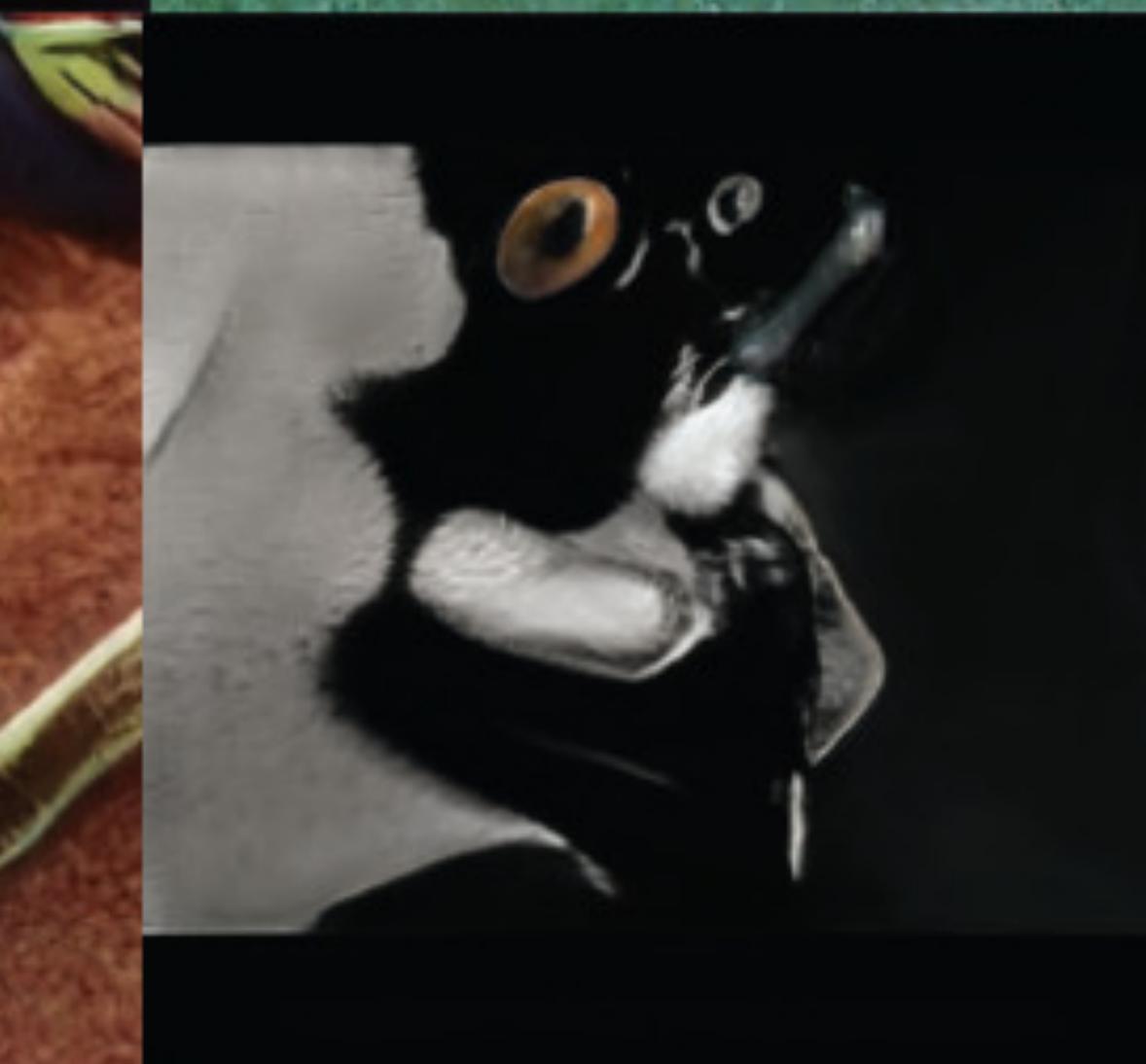


Examples
(intractable likelihood $p_t(x)$)



[Goodfellow et al. 1406.2661]

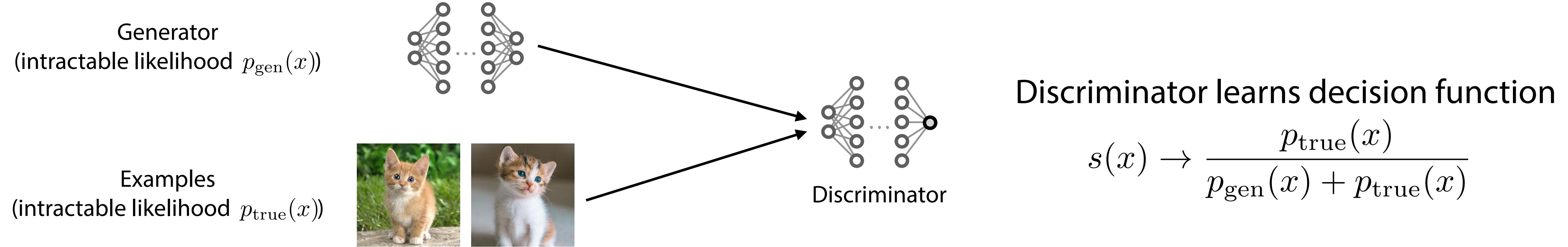
$$\text{decision function} = \frac{p_{\text{true}}(x)}{p_{\text{true}}(x) + p_{\text{true}}(x)}$$



[Nvidia, Universal Pictures]

Idea 1: the likelihood ratio trick

- Generative Adversarial Networks (GANs):

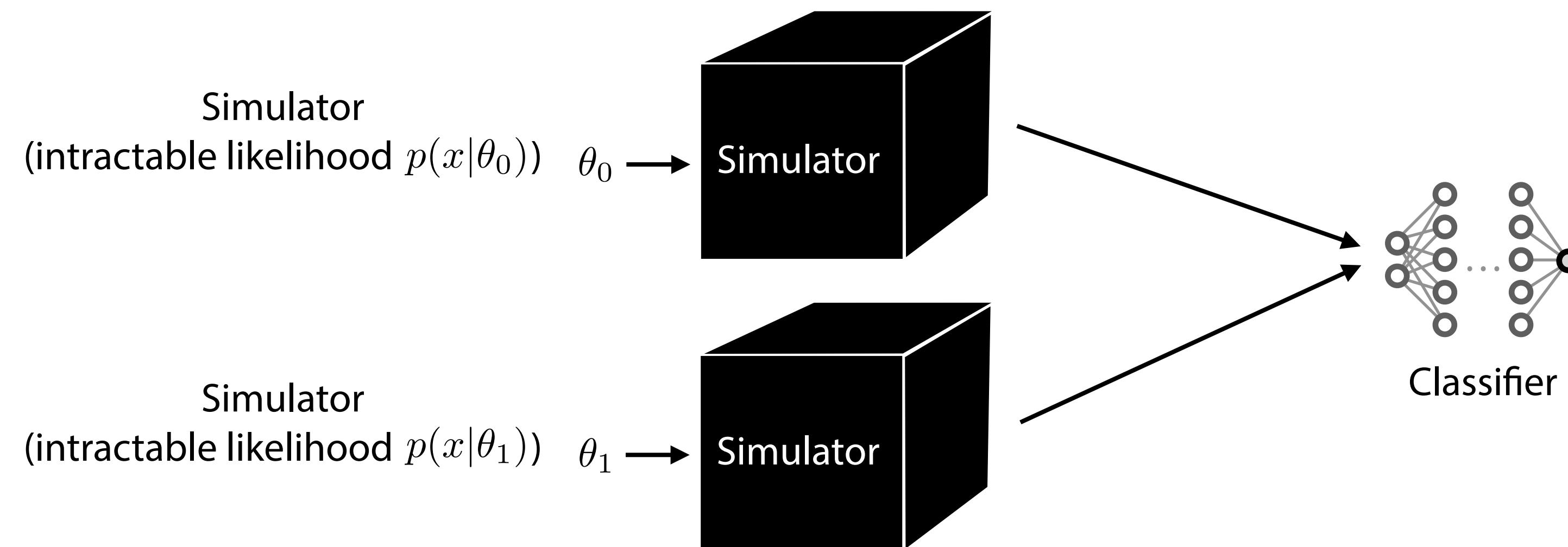


[I. Goodfellow et al. 1406.2661]

Discriminator learns decision function

$$s(x) \rightarrow \frac{p_{\text{true}}(x)}{p_{\text{gen}}(x) + p_{\text{true}}(x)}$$

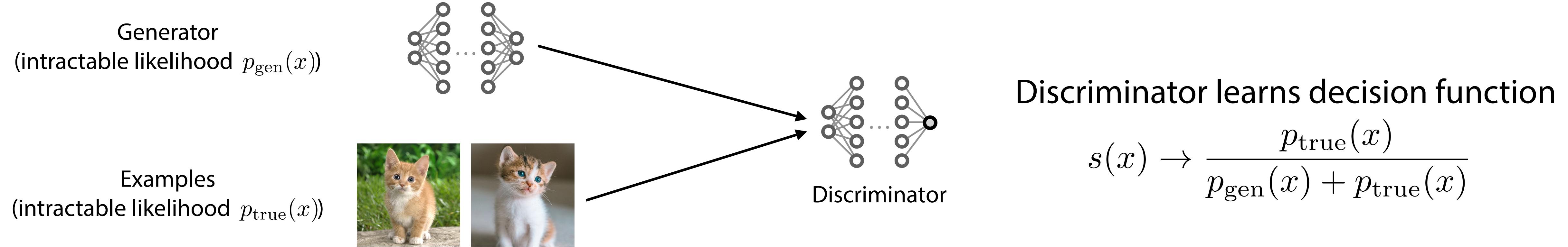
- Similarly, we can train a classifier between two sets of simulated samples



[K. Cranmer, J. Pavez, G. Louppe 1506.02169]

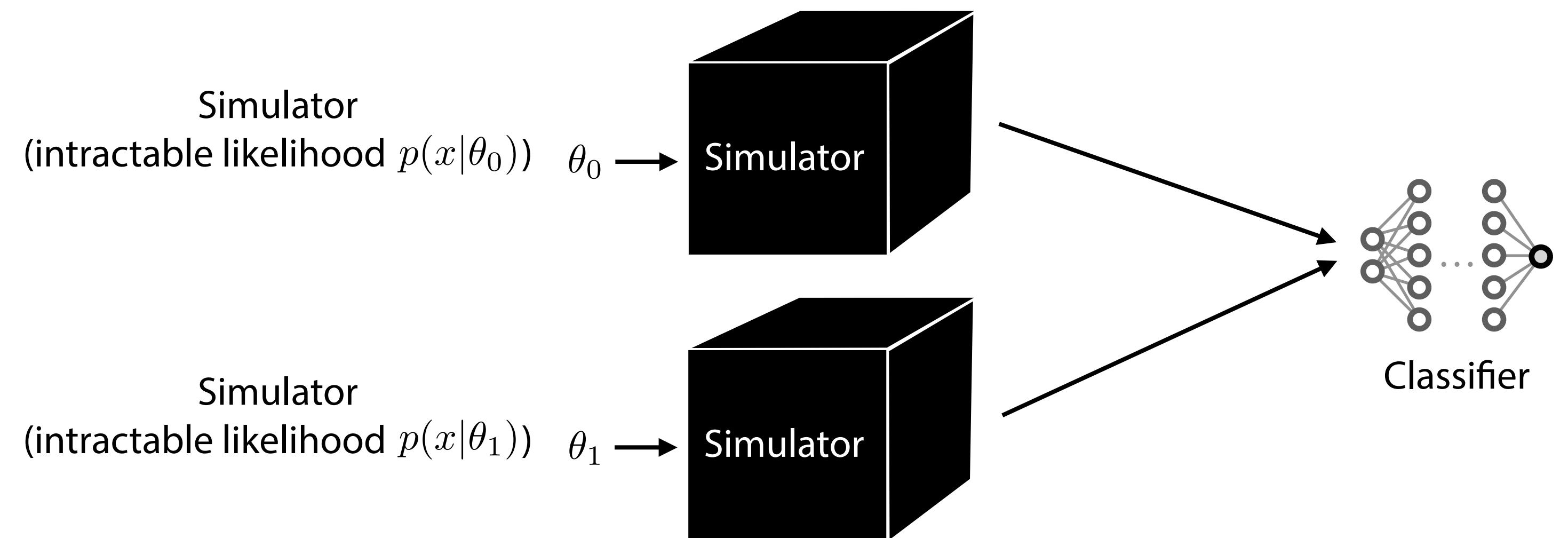
Idea 1: the likelihood ratio trick

- Generative Adversarial Networks (GANs):



[I. Goodfellow et al. 1406.2661]

- Similarly, we can train a classifier between two sets of simulated samples



Discriminator learns decision function

$$s(x) \rightarrow \frac{p_{\text{true}}(x)}{p_{\text{gen}}(x) + p_{\text{true}}(x)}$$

[K. Cranmer, J. Pavez, G. Louppe 1506.02169]

Classifier learns decision function

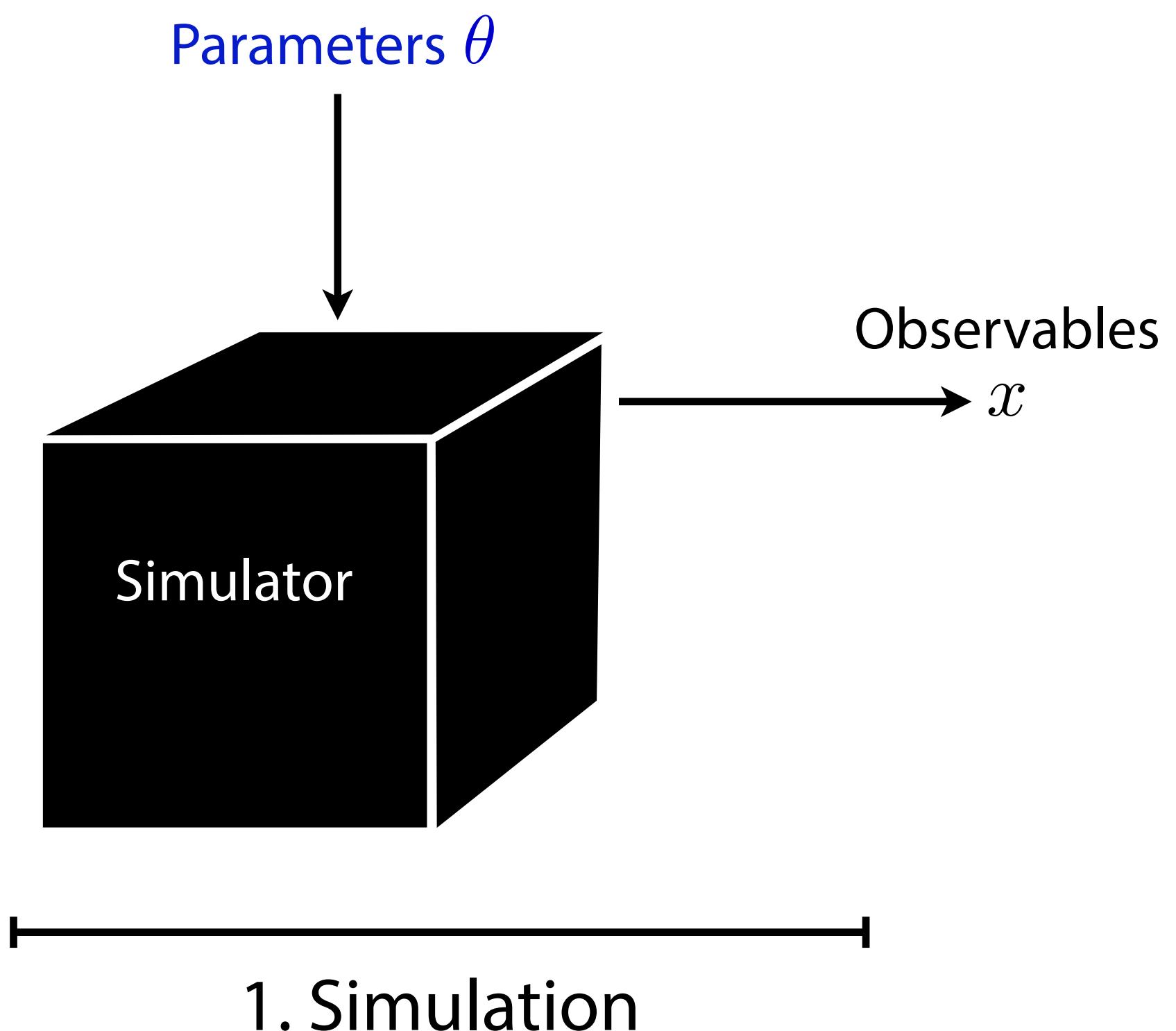
$$s(x) \rightarrow \frac{p(x|\theta_1)}{p(x|\theta_0) + p(x|\theta_1)}$$

⇒ Estimator for likelihood ratio

$$\hat{r}(x) = \frac{1 - s(x)}{s(x)} \rightarrow \frac{p(x|\theta_0)}{p(x|\theta_1)}$$

Inference by likelihood ratio trick

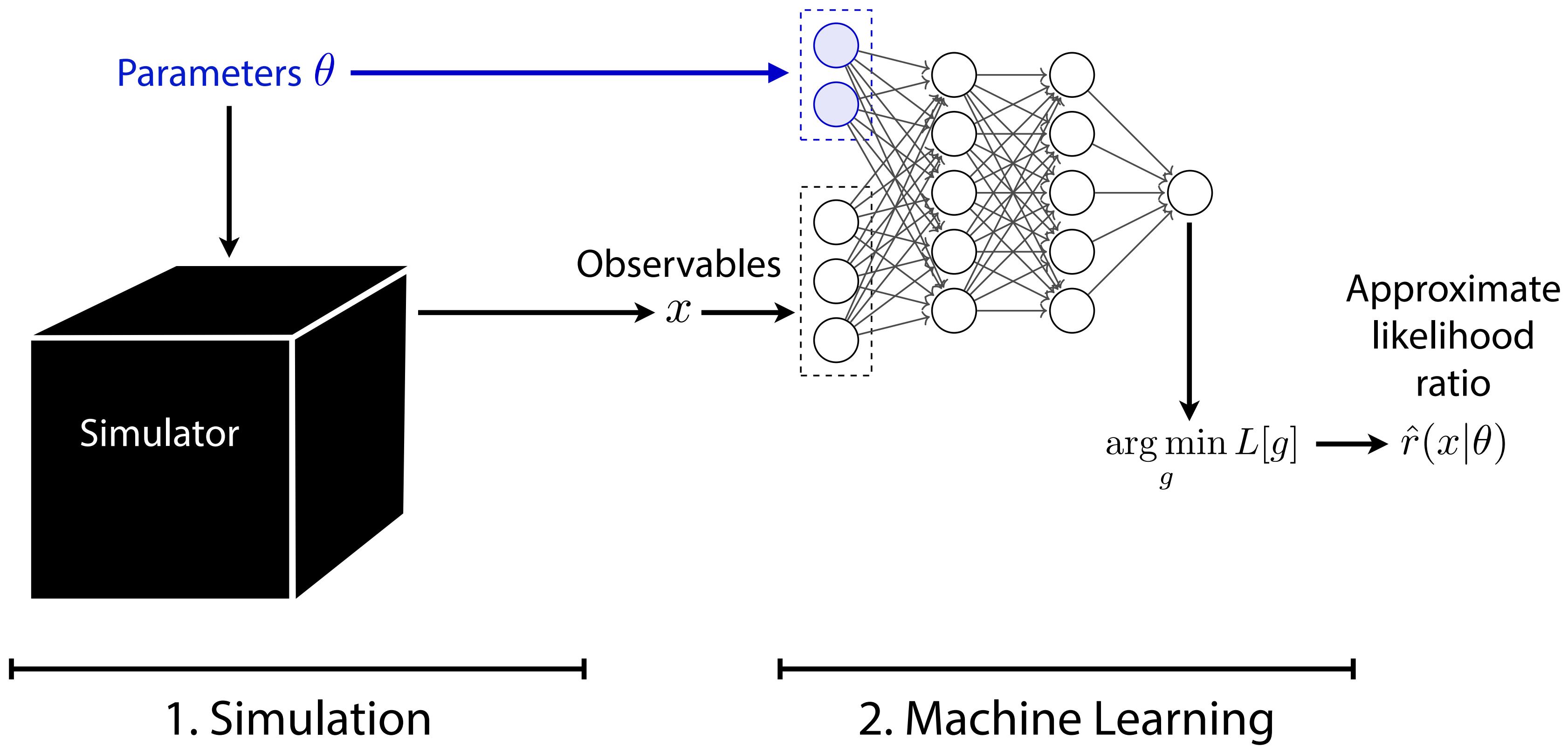
[K. Cranmer, J. Pavez, G. Louppe 1506.02169]



Run simulator and save data

Inference by likelihood ratio trick

[K. Cranmer, J. Pavez, G. Louppe 1506.02169]

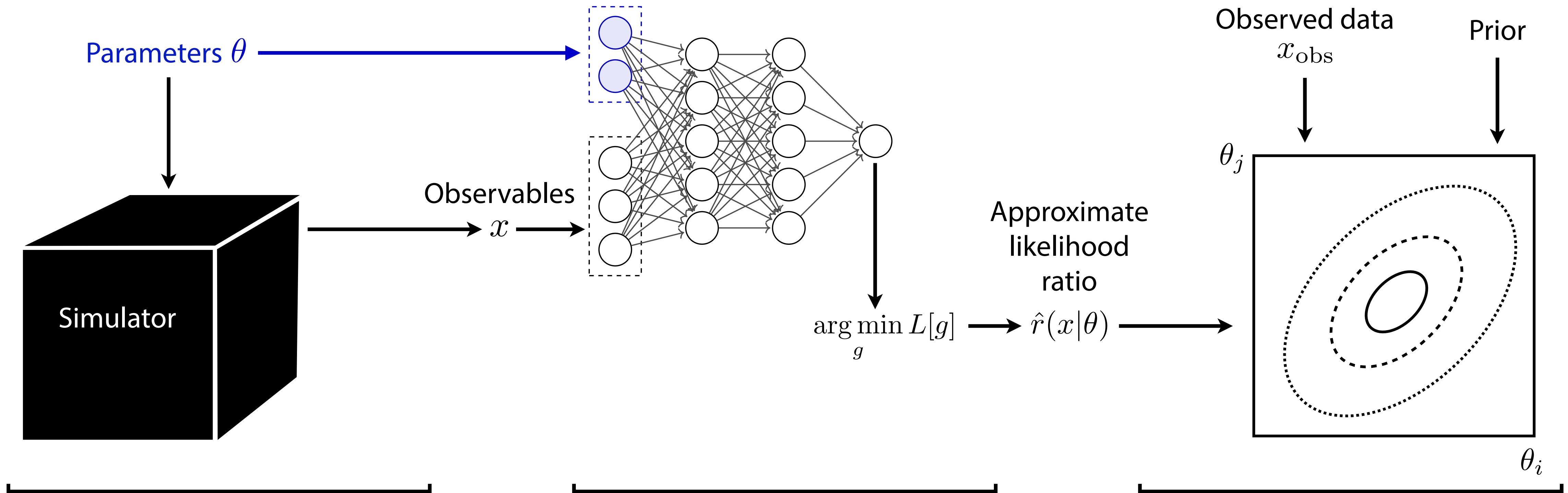


Run simulator and save data

Train NN classifier, interpret as likelihood ratio estimator

Inference by likelihood ratio trick

[K. Cranmer, J. Pavez, G. Louppe 1506.02169]



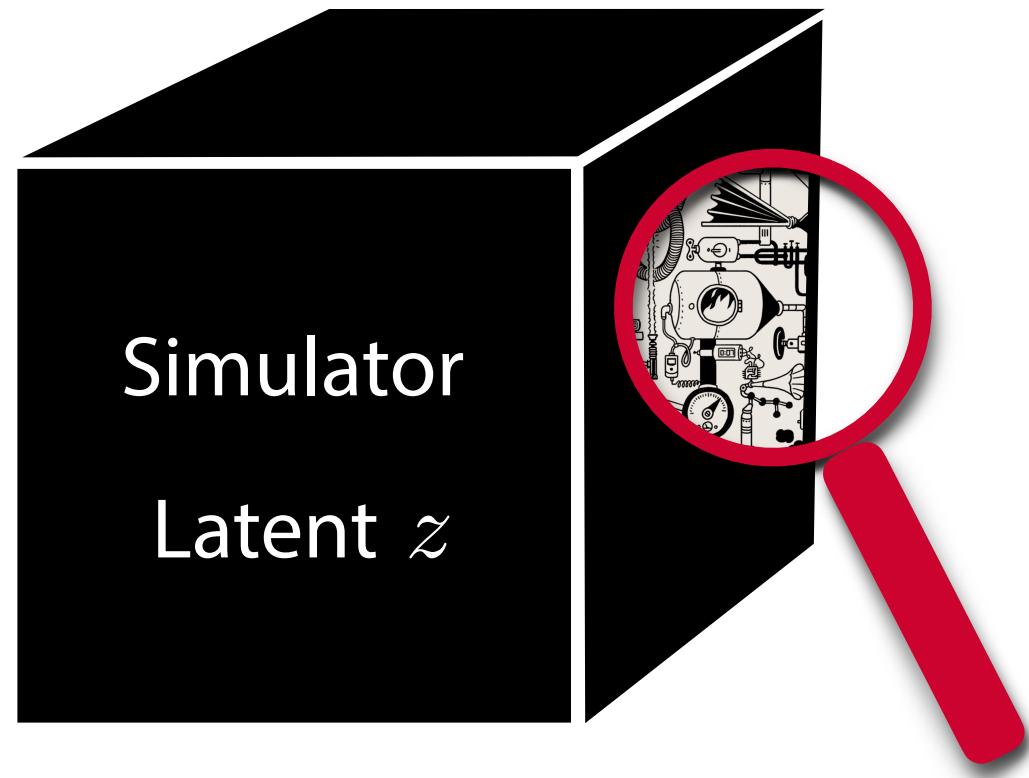
Run simulator and save data

Train NN classifier, interpret as likelihood ratio estimator

Amortized: cheap to repeat for new data

Idea 2: gold mining

[JB, G. Louppe, J. Pavez, K. Cranmer 1805.12244, 1805.00013, 1805.00020]



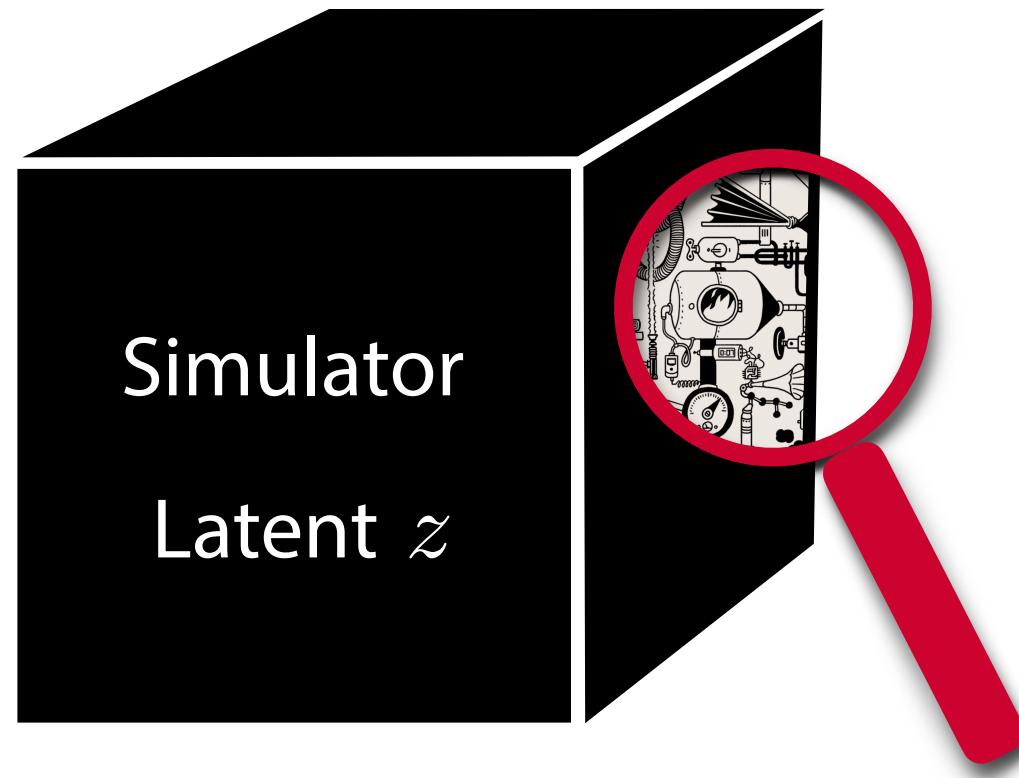
We cannot compute $p(x|\theta) = \int dz p(x, z|\theta)$,
but often we can use domain knowledge (or
probabilistic programming methods) to compute

- the **joint likelihood ratio** $r(x, z|\theta) = \frac{p(x, z|\theta)}{p_{\text{ref}}(x, z)}$
- the **joint score** $t(x, z|\theta) = \nabla_{\theta} \log p(x, z|\theta)$

(Both depend on the simulator latent variables z)

Idea 2: gold mining

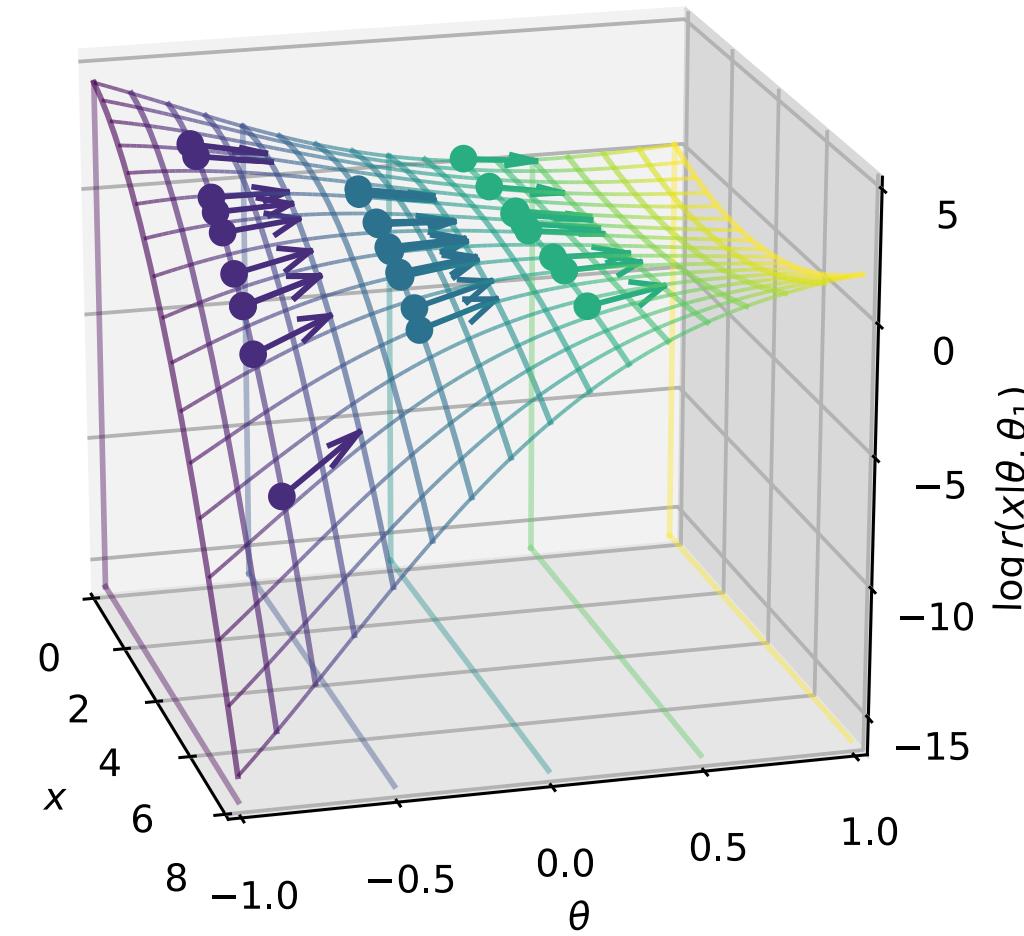
[JB, G. Louppe, J. Pavez, K. Cranmer 1805.12244, 1805.00013, 1805.00020]



We cannot compute $p(x|\theta) = \int dz p(x, z|\theta)$,
but often we can use domain knowledge (or
probabilistic programming methods) to compute

- the **joint likelihood ratio** $r(x, z|\theta) = \frac{p(x, z|\theta)}{p_{\text{ref}}(x, z)}$
- the **joint score** $t(x, z|\theta) = \nabla_{\theta} \log p(x, z|\theta)$

(Both depend on the simulator latent variables z)



Pleasant surprises: we have shown that

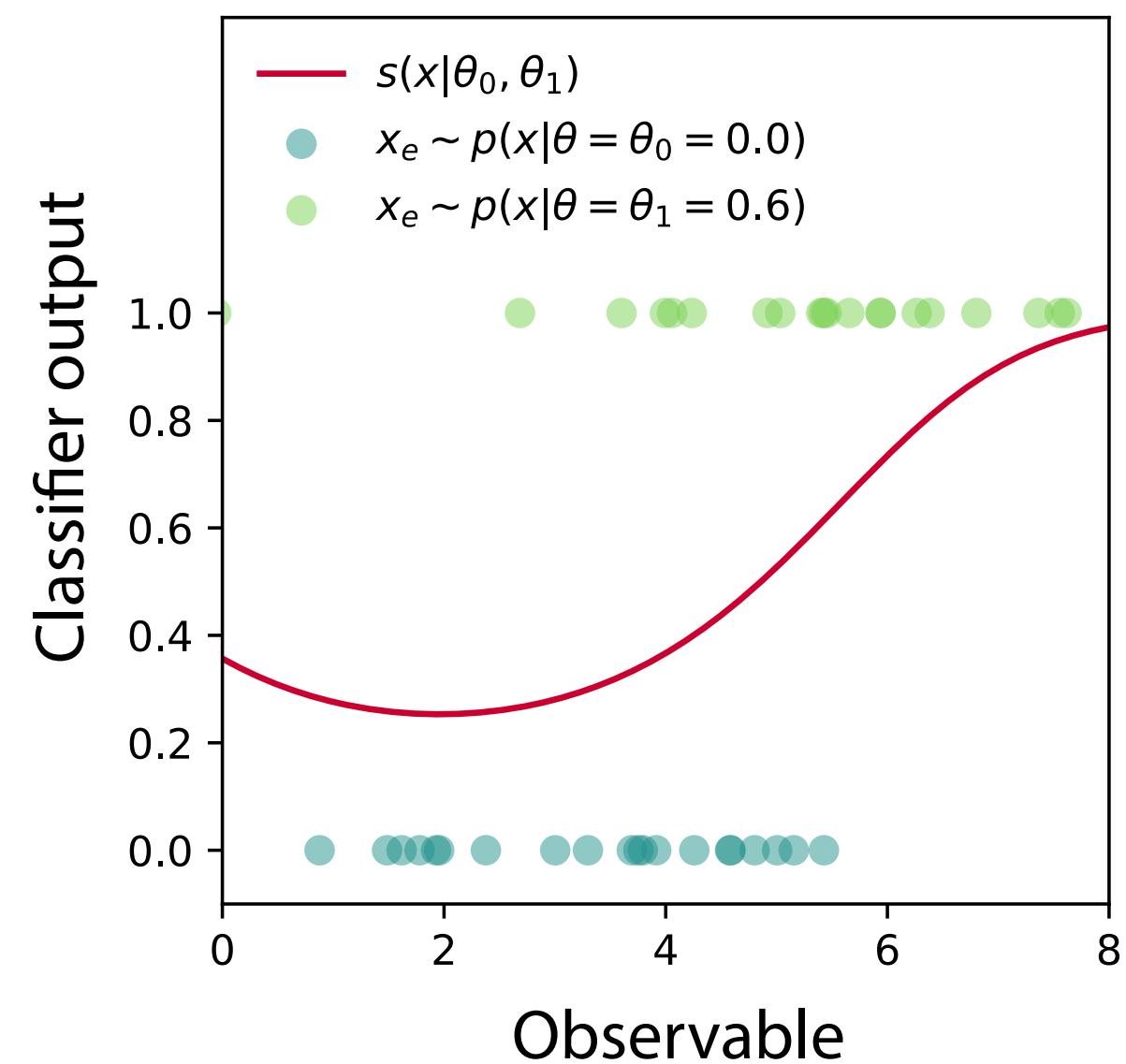
- the **joint likelihood ratio** is an unbiased estimator of the likelihood ratio
- the **joint score** provides unbiased gradient information

⇒ use them as labels in supervised NN training!

Mining gold adds information

[JB, G. Louppe, J. Pavez, K. Cranmer
1805.12244, 1805.00013, 1805.00020]

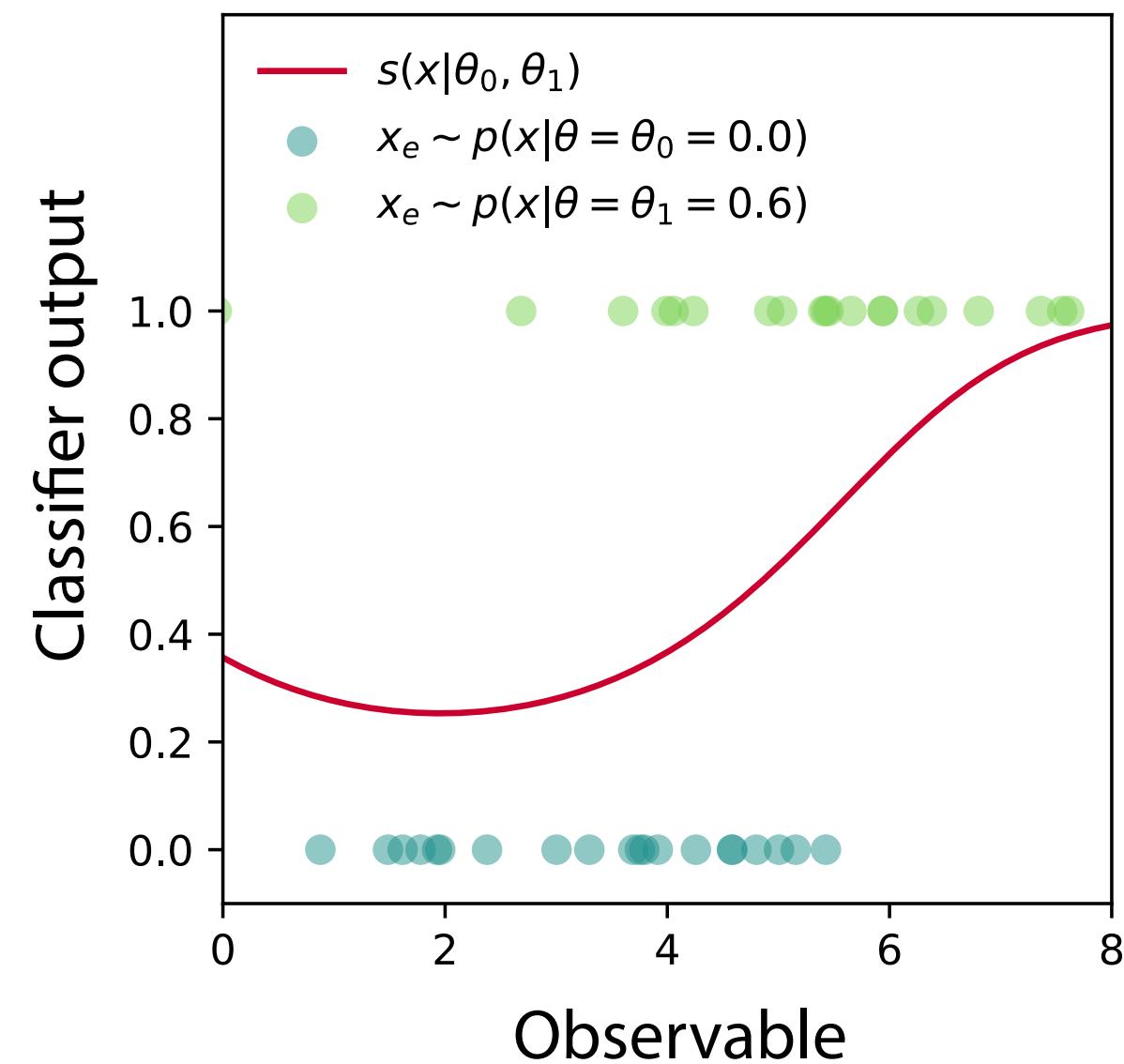
Likelihood ratio trick



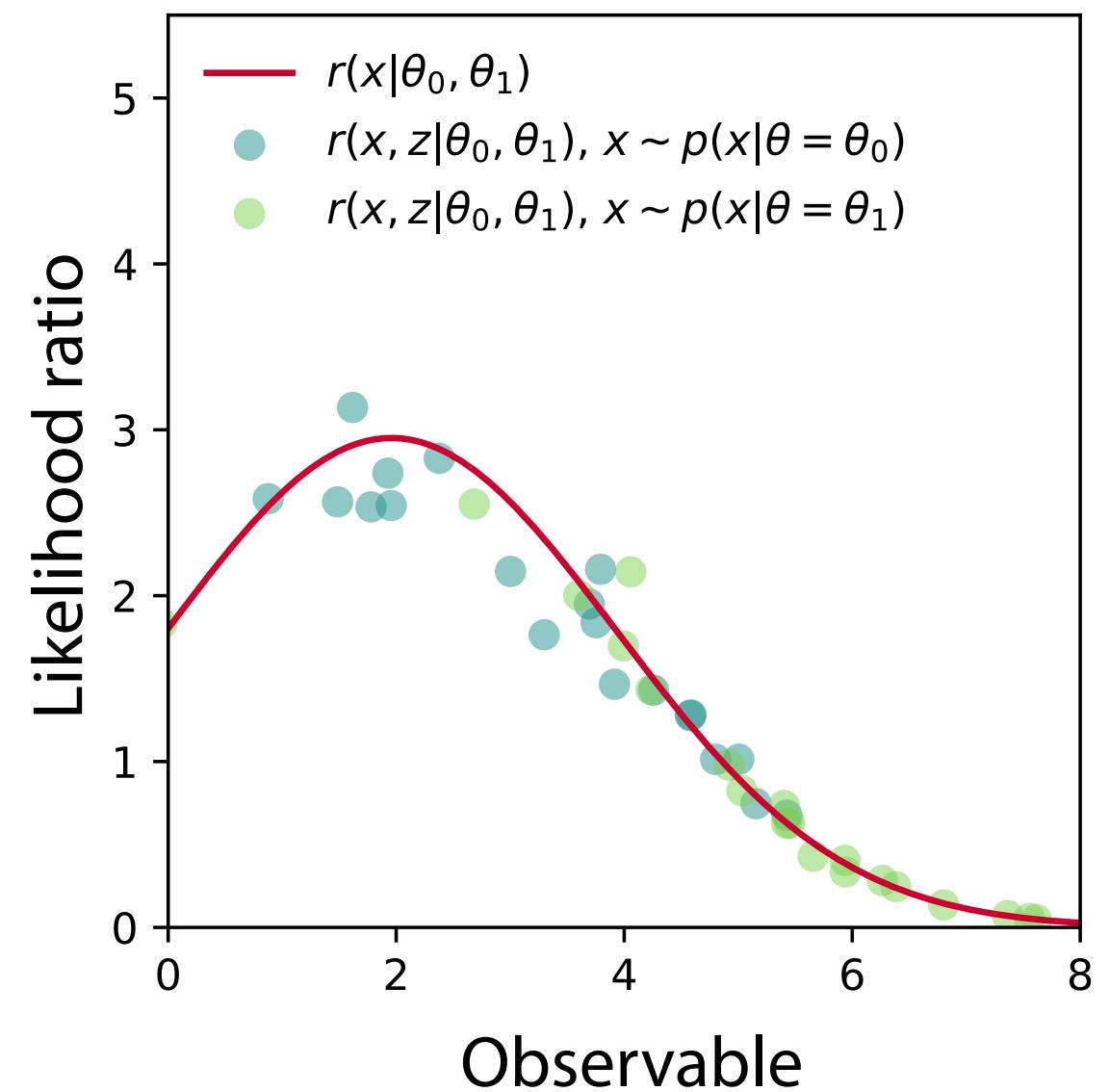
Mining gold adds information

[JB, G. Louppe, J. Pavez, K. Cranmer
1805.12244, 1805.00013, 1805.00020]

Likelihood ratio trick



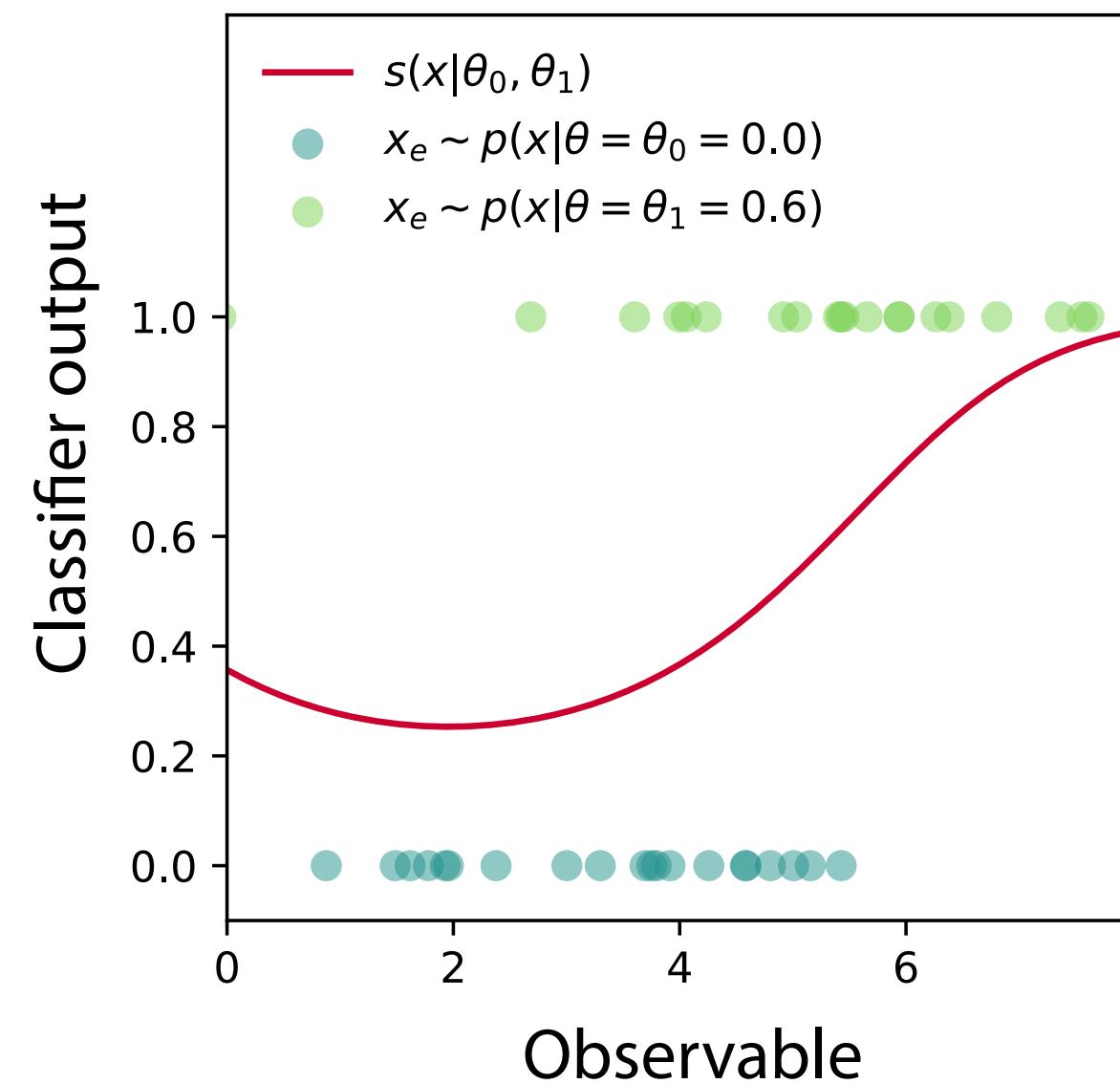
+ joint likelihood ratio



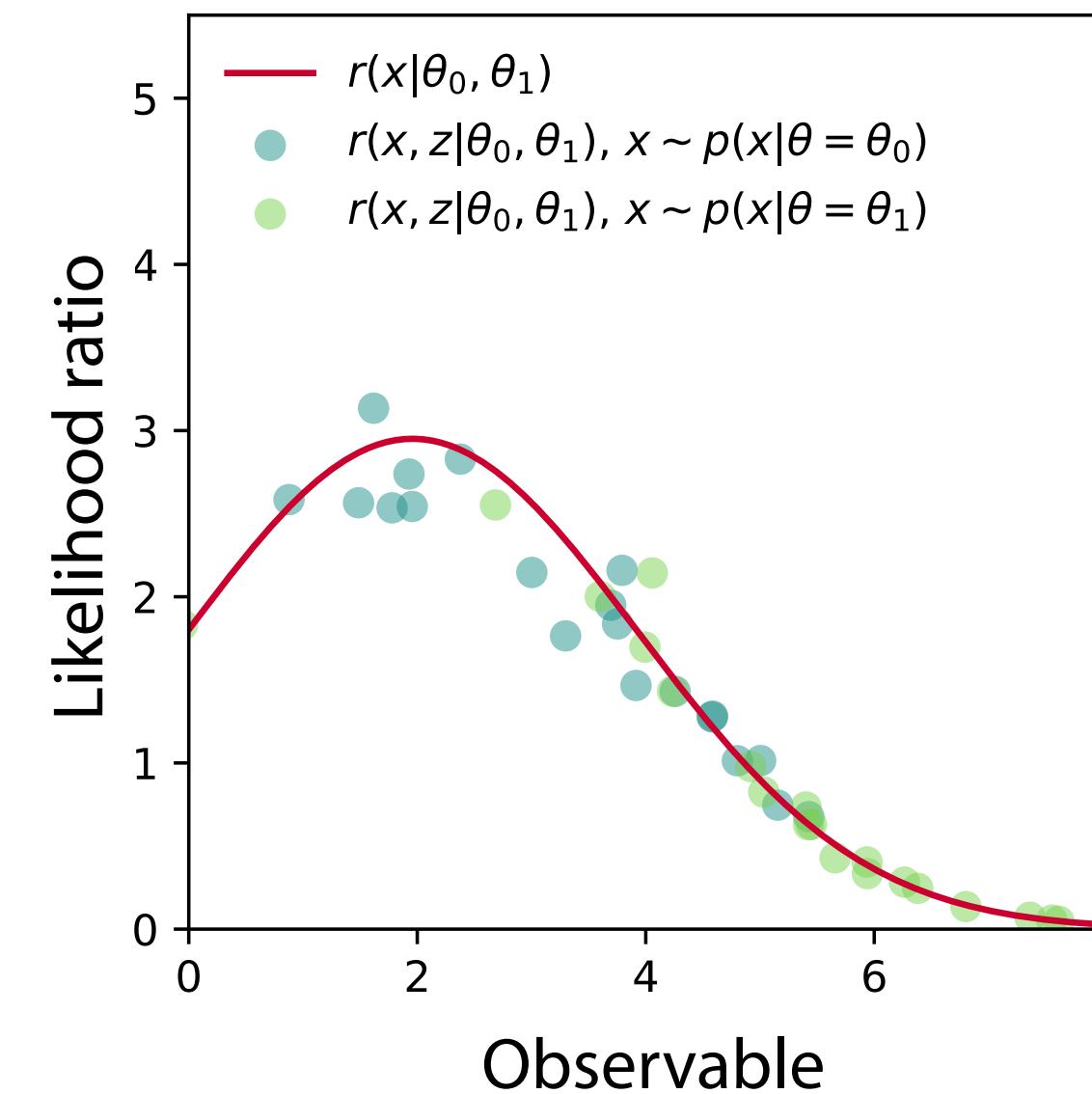
Mining gold adds information

[JB, G. Louppe, J. Pavez, K. Cranmer
1805.12244, 1805.00013, 1805.00020]

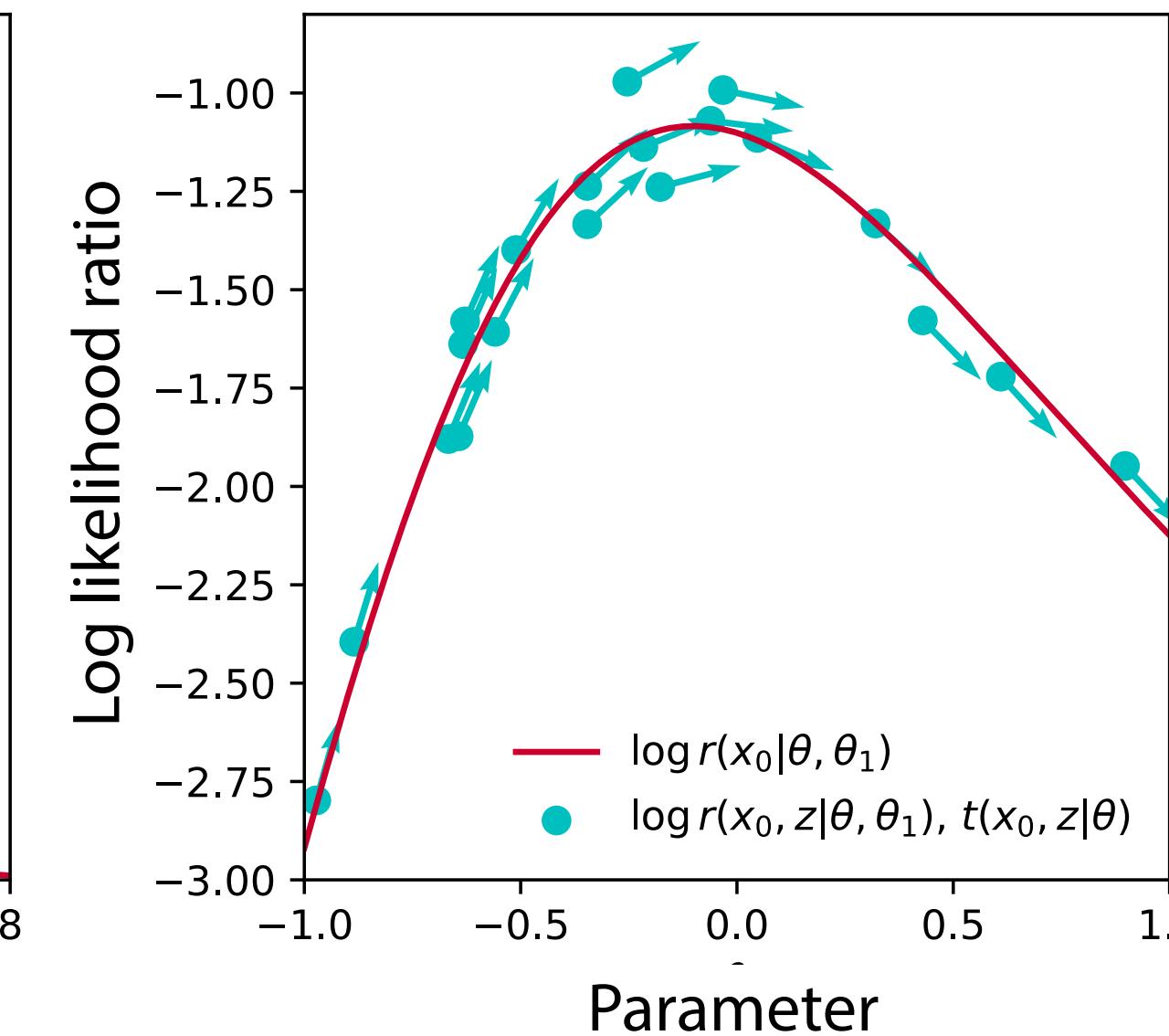
Likelihood ratio trick



+ joint likelihood ratio



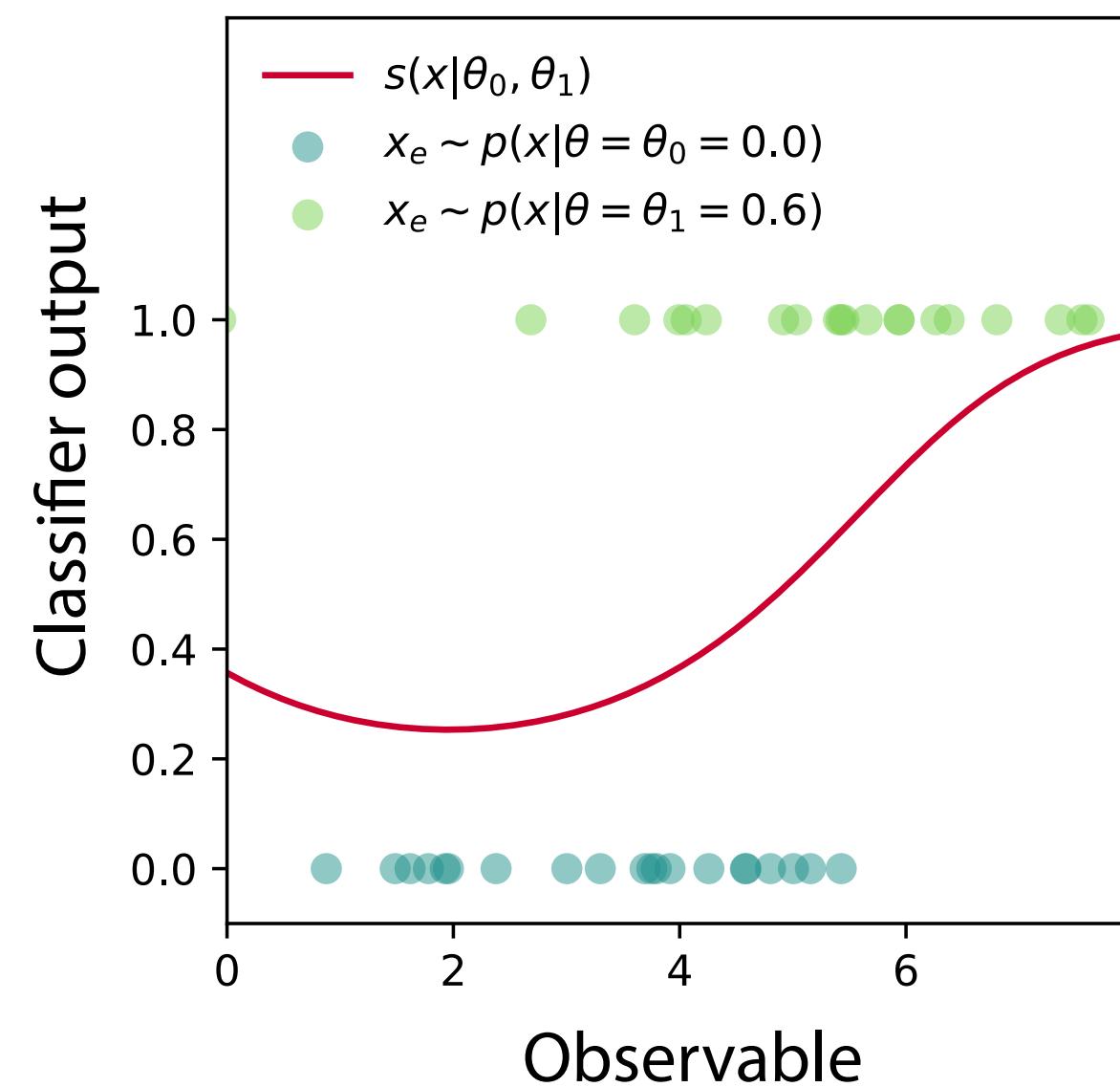
+ joint score



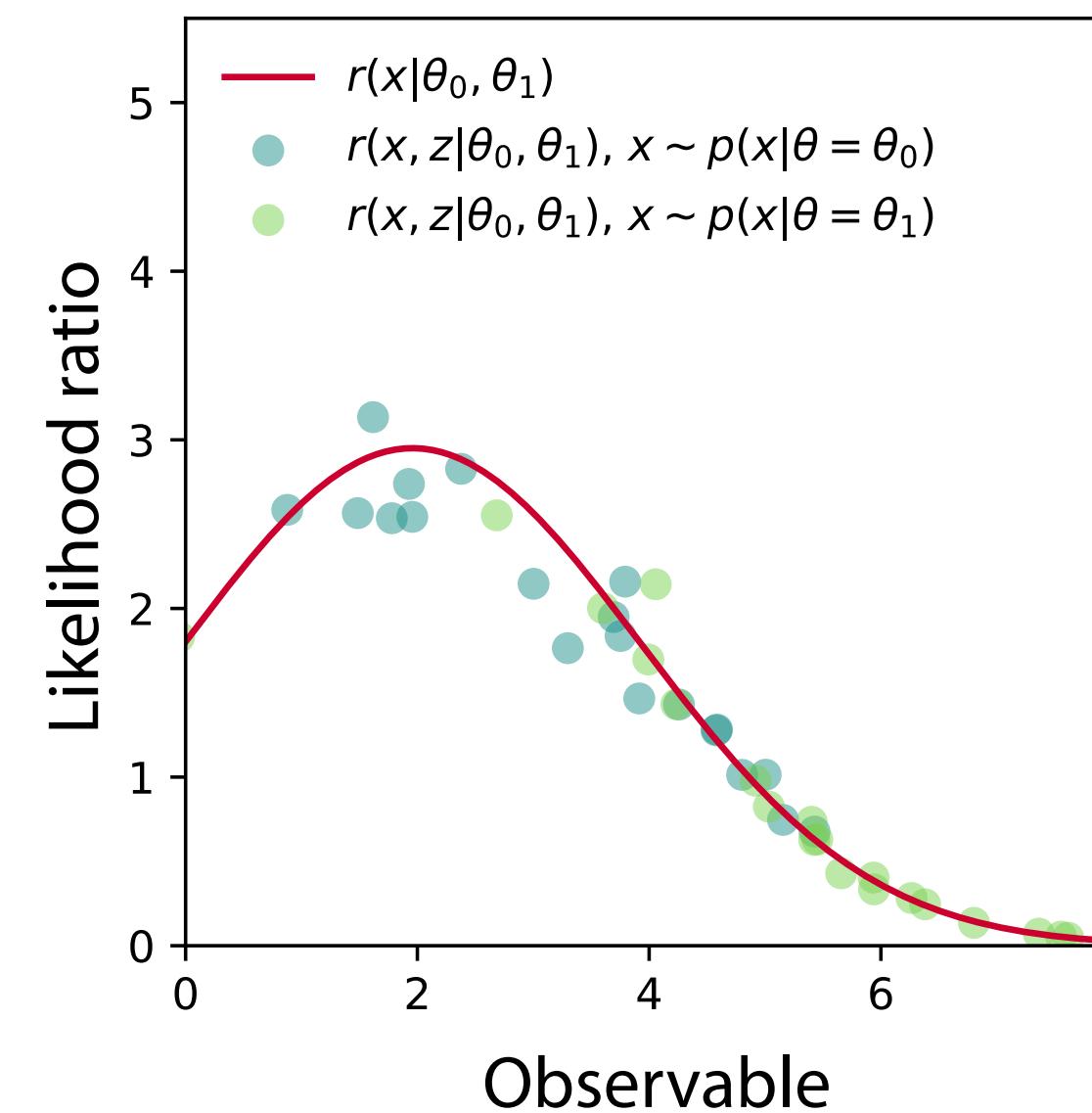
Mining gold adds information

[JB, G. Louppe, J. Pavez, K. Cranmer
1805.12244, 1805.00013, 1805.00020]

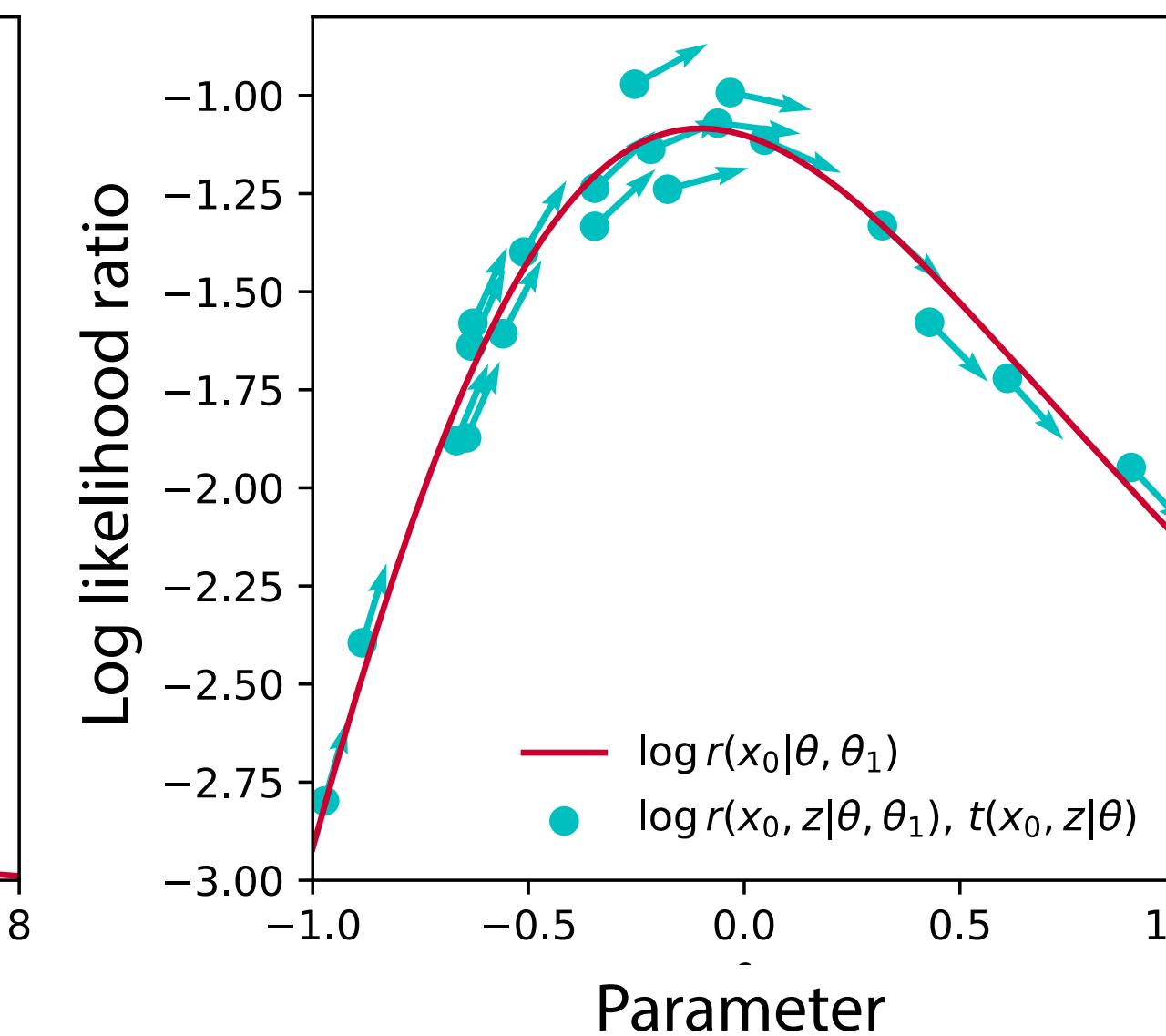
Likelihood ratio trick



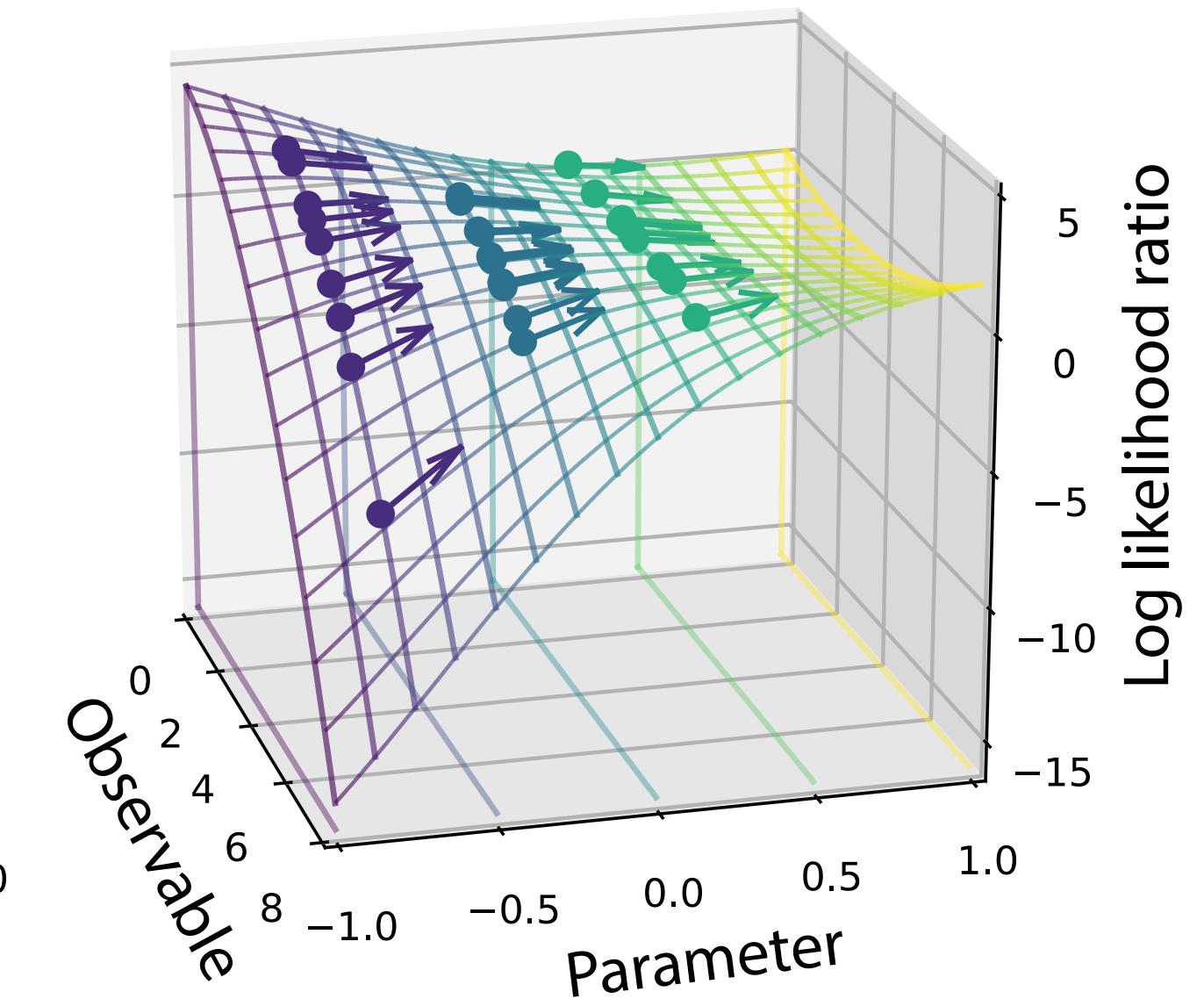
+ joint likelihood ratio



+ joint score



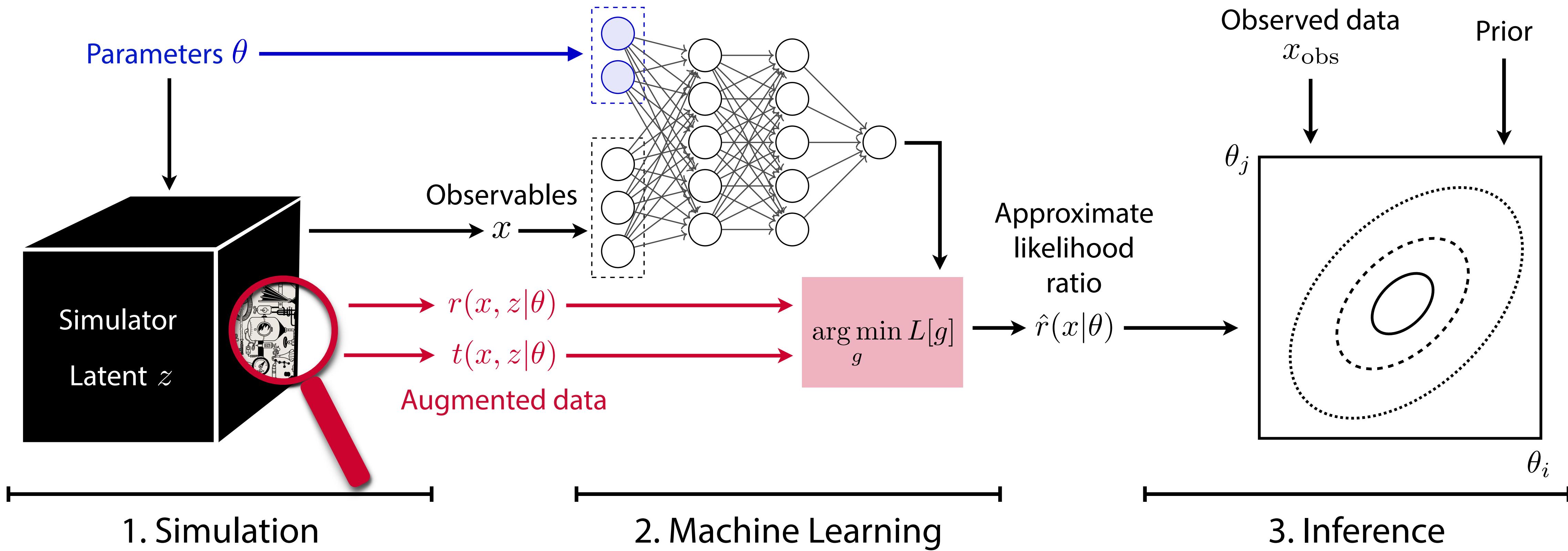
= RASCAL



Using more information = more sample-efficient inference

RASCAL: Likelihood ratio trick + gold mining

[JB, G. Louppe, J. Pavez, K. Cranmer
1805.12244, 1805.00013, 1805.00020]

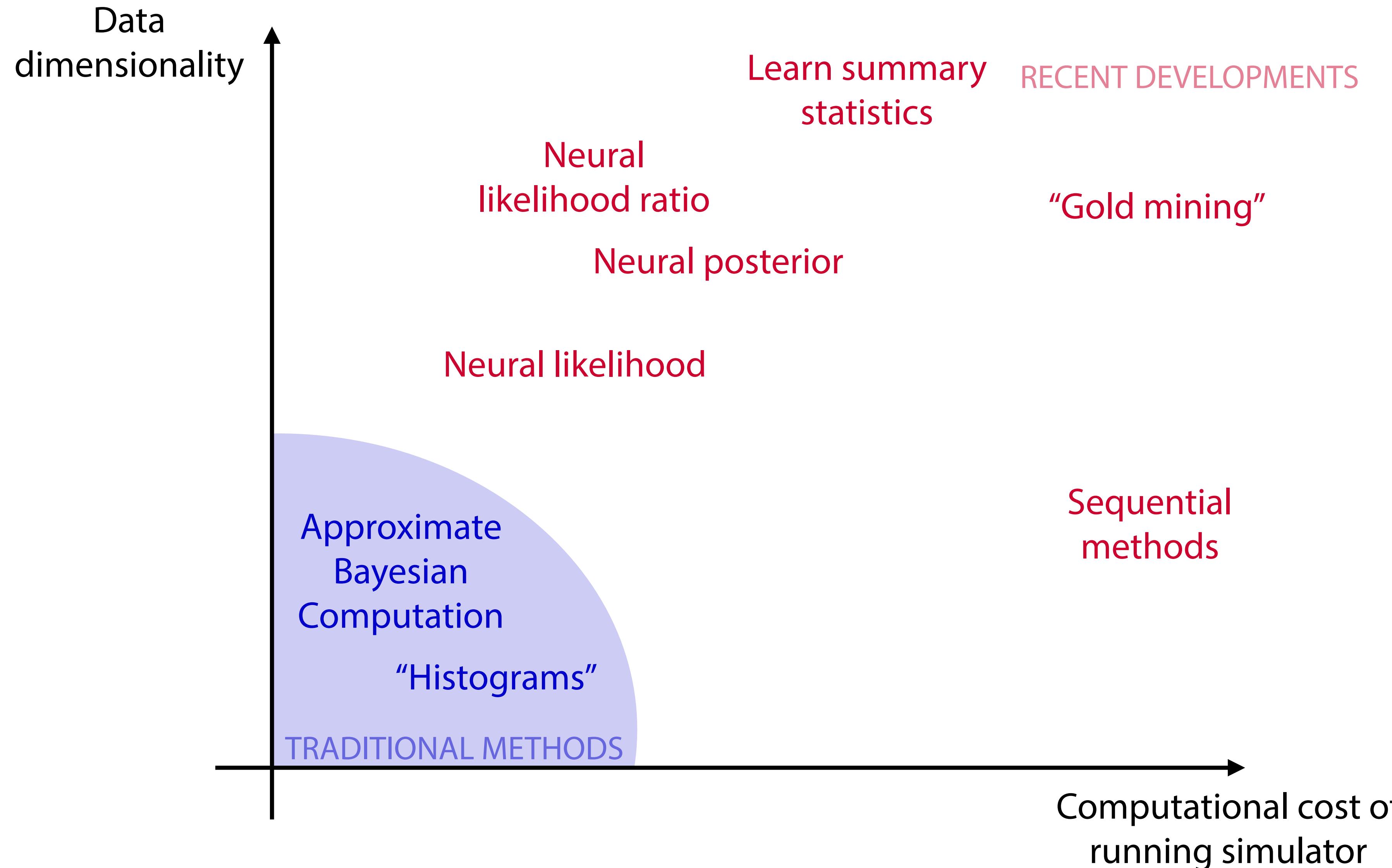


Extract joint likelihood ratio
and joint score from simulator

Augment training data &
use as labels in new loss functions
⇒ improve training efficiency

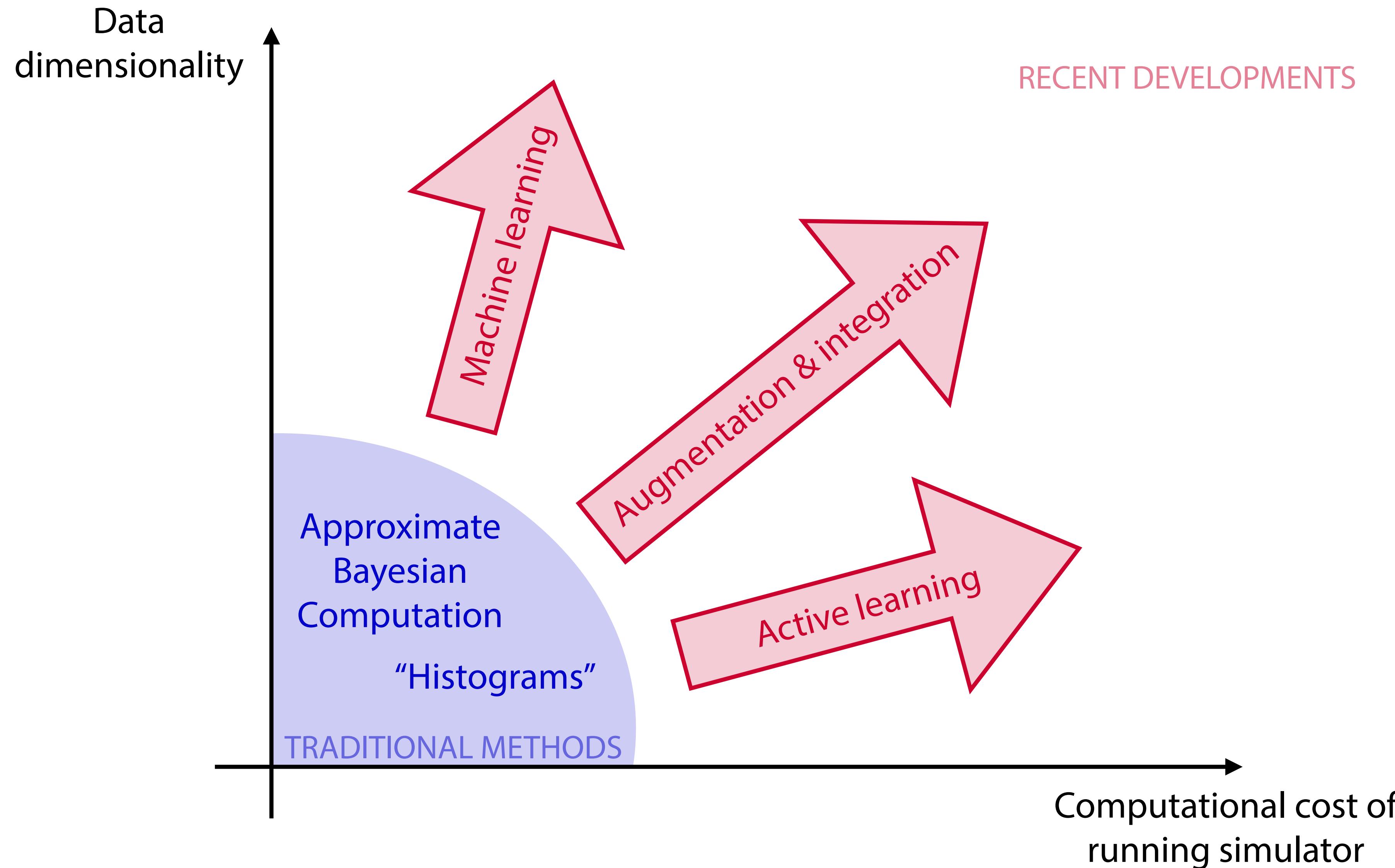
The frontier of simulation-based inference

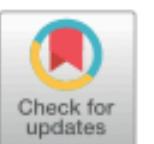
[K. Cranmer, JB, G. Louppe 1911.01429]



The frontier of simulation-based inference

[K. Cranmer, JB, G. Louppe 1911.01429]





The frontier of simulation-based inference

Kyle Cranmer^{a,b,1} , Johann Brehmer^{a,b} , and Gilles Louuppe^c

^aCenter for Cosmology and Particle Physics, New York University, New York, NY 10003; ^bCenter for Data Science, New York University, New York, NY 10011; and ^cMontefiore Institute, University of Liège, B-4000 Liège, Belgium

Edited by Jitendra Malik, University of California, Berkeley, CA, and approved April 10, 2020 (received for review November 4, 2019)

Many domains of science have developed complex simulations to describe phenomena of interest. While these simulations provide high-fidelity models, they are poorly suited for inference and lead to challenging inverse problems. We review the rapidly developing field of simulation-based inference and identify the forces giving additional momentum to the field. Finally, we describe how the frontier is expanding so that a broad audience can appreciate the profound influence these developments may have on science.

statistical inference | implicit models | likelihood-free inference | approximate Bayesian computation | neural density estimation

Mechanistic models can be used to predict how systems will behave in a variety of circumstances. These run the gamut of distance scales, with notable examples including particle physics, molecular dynamics, protein folding, population genetics, neuroscience, epidemiology, economics, ecology, climate science, astrophysics, and cosmology. 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. Unfortunately, these simulators are poorly suited for statistical inference. The source of the challenge is that the probability density (or likelihood) for a given observation—an essential ingredient for both frequentist and Bayesian inference methods—is typically intractable. Such models are often referred to as implicit models and contrasted against prescribed models where the likelihood for an observation can be explicitly calculated (1). The problem setting of statistical inference under intractable likelihoods has been dubbed likelihood-free inference—although it is a bit of a misnomer as typically one attempts to estimate the intractable likelihood, so we feel the term simulation-based inference is more apt.

The intractability of the likelihood is an obstruction for scientific progress as statistical inference is a key component of the scientific method. In areas where this obstruction has appeared, scientists have developed various ad hoc or field-specific methods to overcome it. In particular, two common traditional approaches rely on scientists to use their insight into the system to construct powerful summary statistics and then compare the observed data to the simulated data. In the first one, density estimation methods are used to approximate the distribution of

the simulator—is being recognized as a key idea to improve the sample efficiency of various inference methods. A third direction of research has stopped treating the simulator as a black box and focused on integrations that allow the inference engine to tap into the internal details of the simulator directly.

Amidst this ongoing revolution, the landscape of simulation-based inference is changing rapidly. In this review we aim to provide the reader with a high-level overview of the basic ideas behind both old and new inference techniques. Rather than discussing the algorithms in technical detail, we focus on the current frontiers of research and comment on some ongoing developments that we deem particularly exciting.

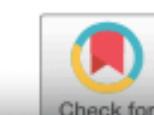
Simulation-Based Inference

Simulators. Statistical inference is performed within the context of a statistical model, and in simulation-based inference the simulator itself defines the statistical model. For the purpose of this paper, a simulator is a computer program that takes as input a vector of parameters θ , samples a series of internal states or latent variables $z_i \sim p_i(z_i|\theta, z_{<i})$, and finally produces a data vector $x \sim p(x|\theta, z)$ as output. Programs that involve random samplings and are interpreted as statistical models are known as probabilistic programs, and simulators are an example. Within this general formulation, real-life simulators can vary substantially:

- The parameters θ describe the underlying mechanistic model and thus affect the transition probabilities $p_i(z_i|\theta, z_{<i})$. Typically the mechanistic model is interpretable by a domain scientist and θ has relatively few components and a fixed dimensionality. Examples include coefficients found in the Hamiltonian of a physical system, the virulence and incubation rate of a pathogen, or fundamental constants of Nature.
- The latent variables z that appear in the data-generating process may directly or indirectly correspond to a physically meaningful state of a system, but typically this state is unobservable in practice. The structure of the latent space varies substantially between simulators. The latent variables may be continuous or discrete and the dimensionality of the latent space may be fixed or may vary, depending on the control flow of the simulator. The simulation can freely combine deterministic and stochastic steps. The deterministic components of the simulator may be differentiable or may involve discontinuous control

RECENT DEVELOPMENTS

Computational cost of running simulator



The frontier of simulation-based inference

Kyle Cranmer^{a,b,1} , Johann Brehmer^{a,b} , and Gilles Louuppe^c

^aCenter for Cosmology and Particle Physics, New York University, New York, NY 10003; ^bCenter for Data Science, New York University, New York, NY 10003; ^cMontefiore Institute, University of Liège, B-4000 Liège, Belgium

Edited by Jitendra Malik, University of California, Berkeley, CA, and approved April 10, 2020 (received for review November 1, 2019)

Many domains of science have developed complex simulations to describe phenomena of interest. While these simulations provide high-fidelity models, they are poorly suited for inference and lead to challenging inverse problems. We review the rapidly developing field of simulation-based inference and identify the forces giving additional momentum to the field. Finally, we describe how the frontier is expanding so that a broad audience can appreciate the profound influence these developments may have on science.

statistical inference | implicit models | likelihood-free inference | approximate Bayesian computation | neural density estimation

Mechanistic models can be used to predict how systems will behave in a variety of circumstances. These run the gamut of distance scales, with notable examples including particle physics, molecular dynamics, protein folding, population genetics, neuroscience, epidemiology, economics, ecology, climate science, astrophysics, and cosmology. 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. Unfortunately, these simulators are poorly suited for statistical inference. The source of the challenge is that the probability density (or likelihood) for a given observation—an essential ingredient for both frequentist and Bayesian inference methods—is typically intractable. Such models are often referred to as implicit models and contrasted against prescribed models where the likelihood for an observation can be explicitly calculated (1). The problem setting of statistical inference under intractable likelihoods has been dubbed likelihood-free inference—although it is a bit of a misnomer as typically one attempts to estimate the intractable likelihood, so we feel the term simulation-based inference is more apt.

The intractability of the likelihood is an obstruction for scientific progress as statistical inference is a key component of the scientific method. In areas where this obstruction has appeared, scientists have developed various ad hoc or field-specific methods to overcome it. In particular, two common traditional approaches rely on scientists to use their insight into the system to construct powerful summary statistics and then compare the observed data to the simulated data. In the first one, density estimation methods are used to approximate the distribution of

Likelihood-Free Inference Workshop

18-22 March 2019 @ Flatiron Institute, NYC

[Home](#) [Schedule](#) [Hackathon](#) [Logistics](#) [Participants](#) [Registration](#)

Rationale

The goal of this interdisciplinary meeting is to gather developers and users of Likelihood-Free Inference methods to share latest techniques, use cases and applications across different fields, and discuss open challenges.

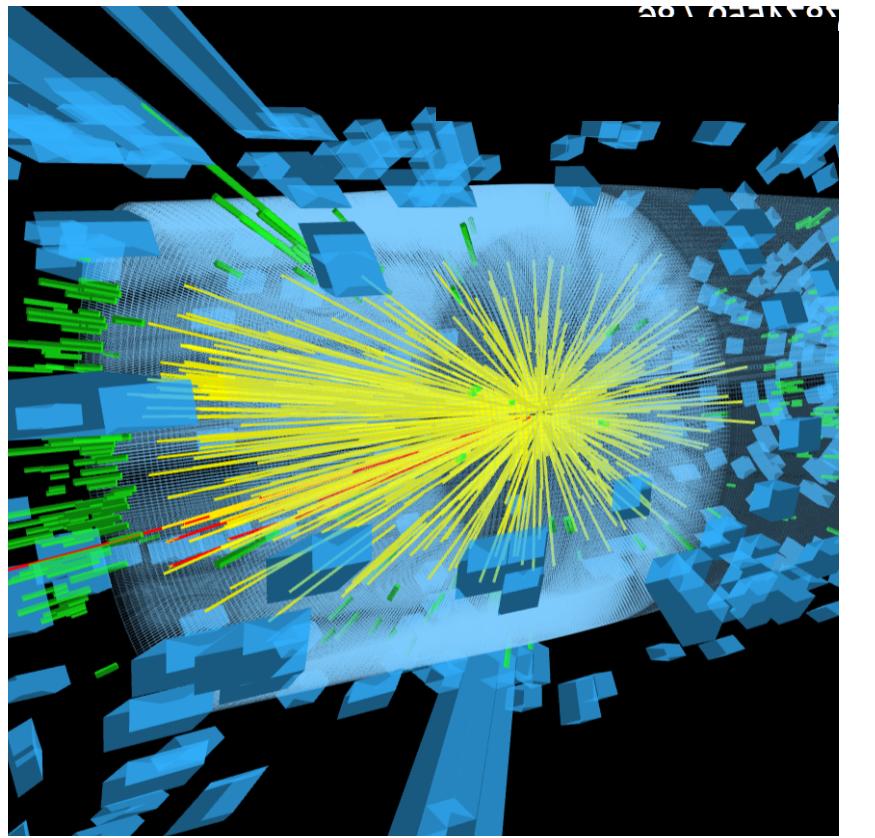
The first two days of the workshop will be focused on talks and discussions, while the remaining days of the week will be dedicated to a hackathon with the goal of seeding the development of a common probabilistic programming framework for Likelihood-Free Inference as well as collaboratively working on LFI-related hack projects.

News

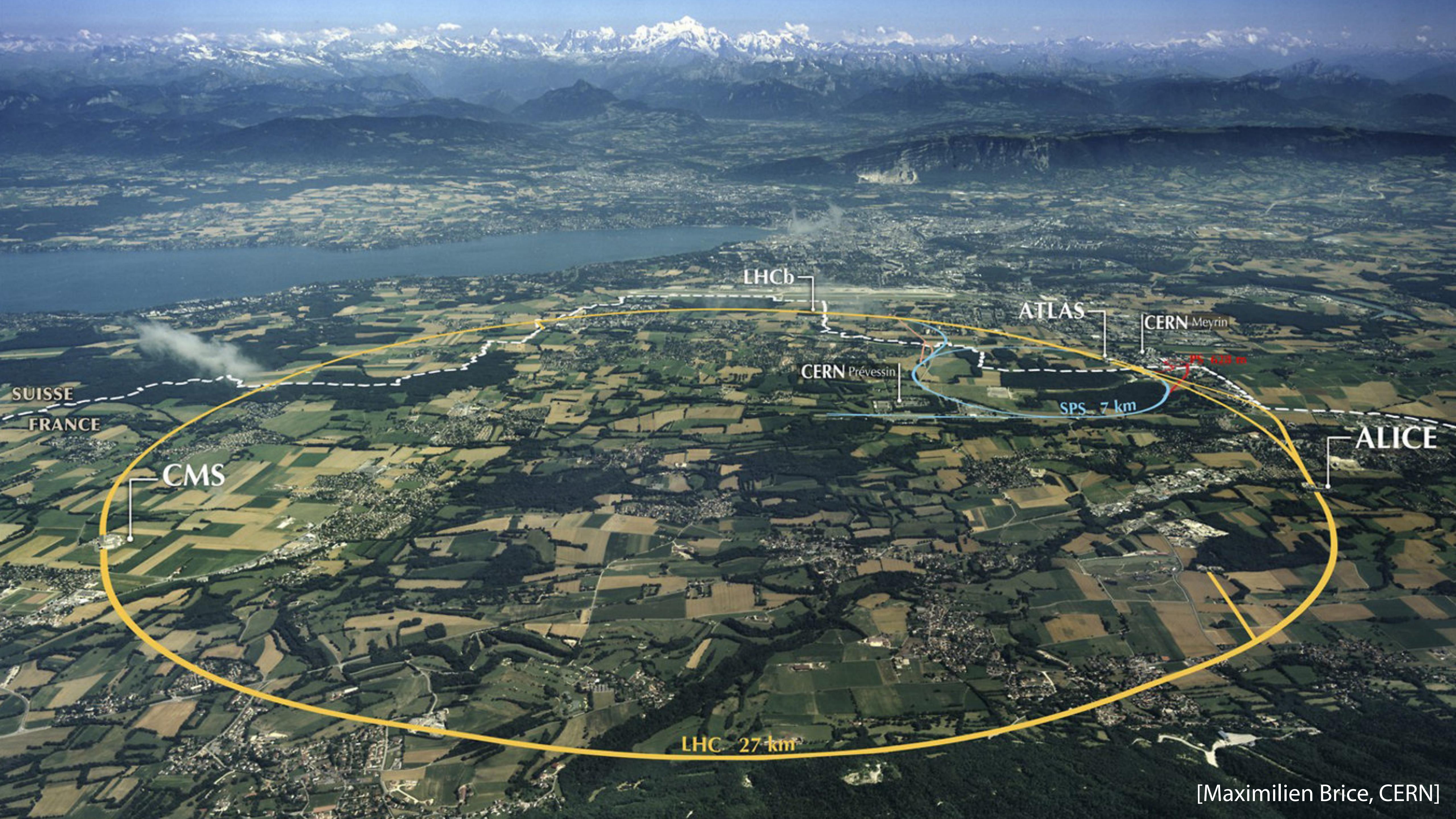
- *March 4th, 2019* : Preliminary schedule [available](#), new Gitter channel [chat on gitter](#), new Hackathon page
- *February 19th, 2019* : Main registration is closed, contact organizers for late registration
- *February 19th, 2019* : Travel funding application deadline
- *February 6th, 2019* : Opening registration

Organizing Committee

- [Justin Alsing](#), Oskar Klein Center, Stockholm University
- [Johann Brehmer](#), Center for Data Science, New York University
- [Stephen Feeney](#), Center for Computational Astrophysics, Flatiron Institute



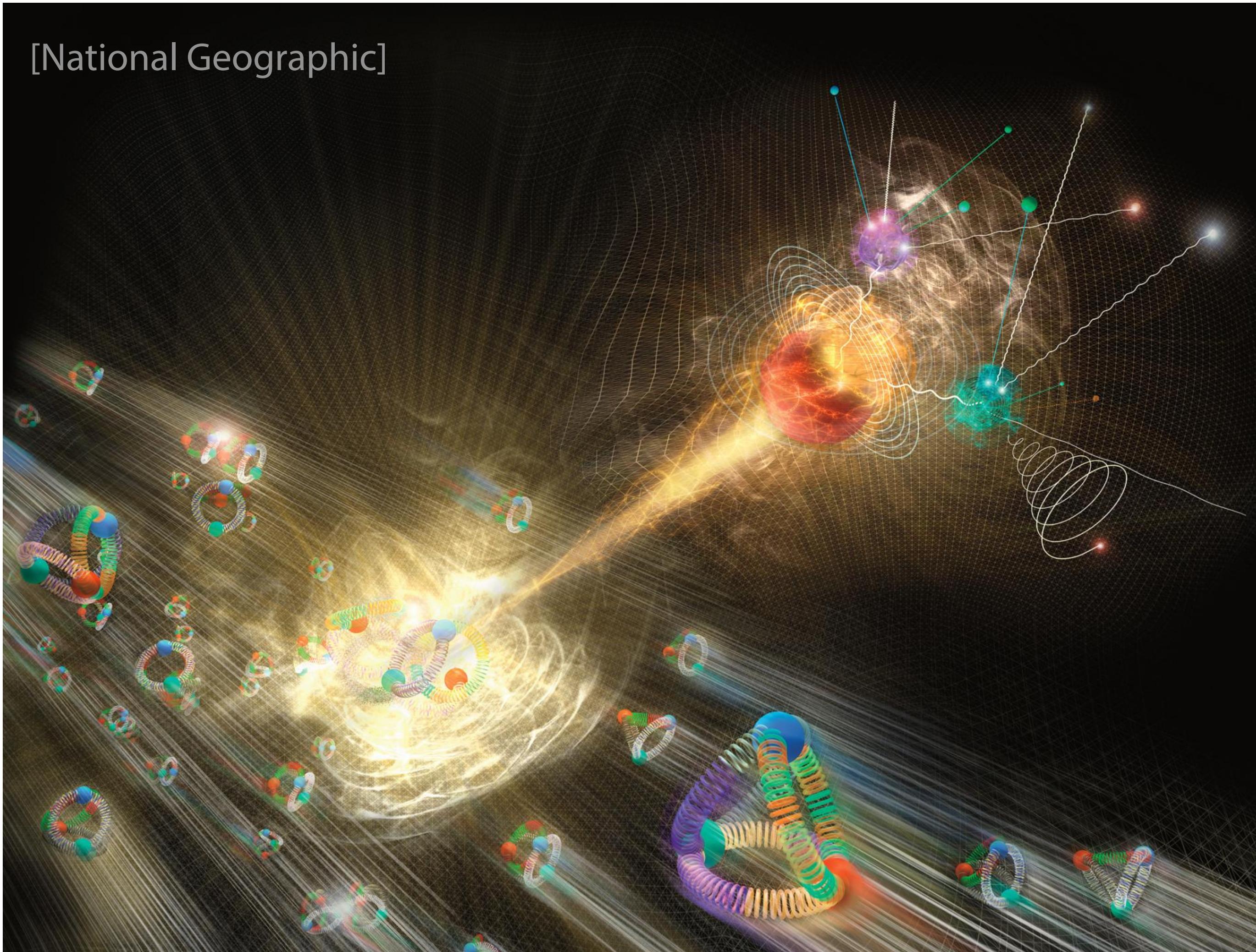
3. Particle physics



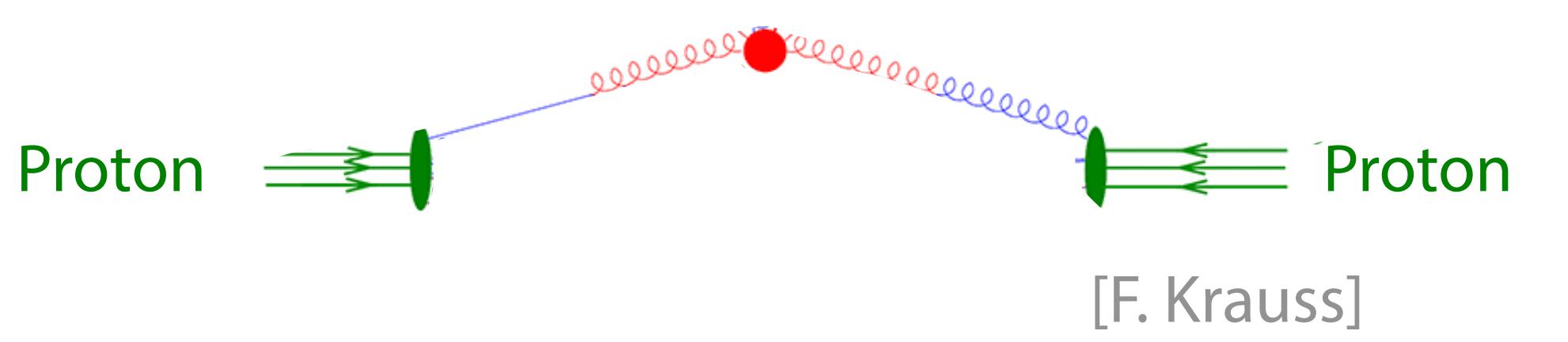
[Maximilien Brice, CERN]

Proton collisions

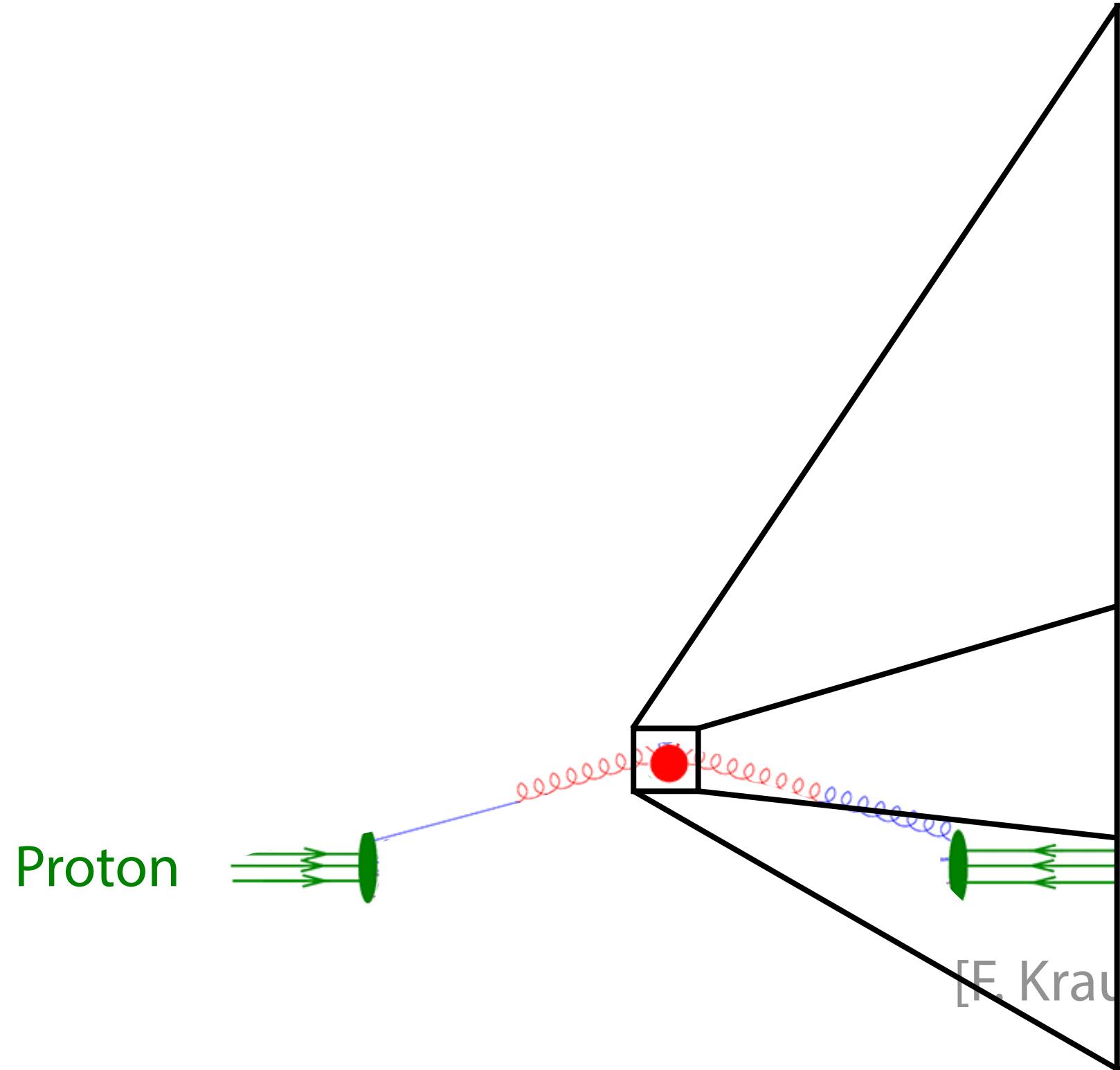
[National Geographic]



Proton collisions

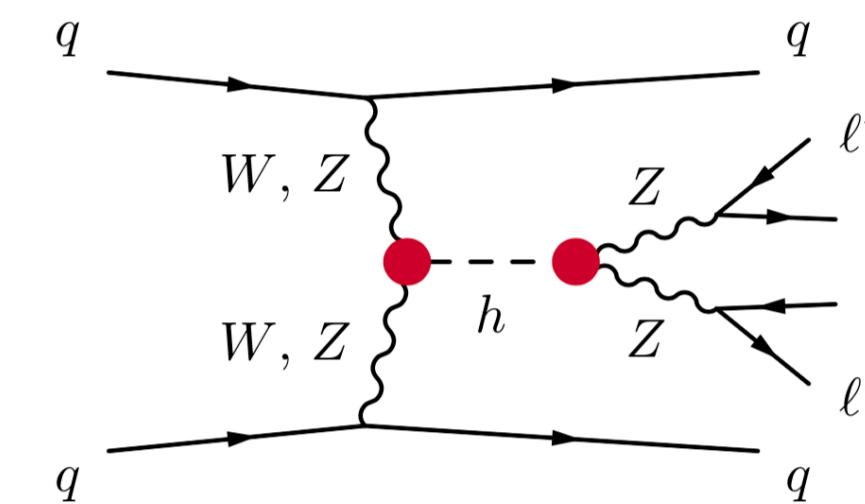


Proton collisions



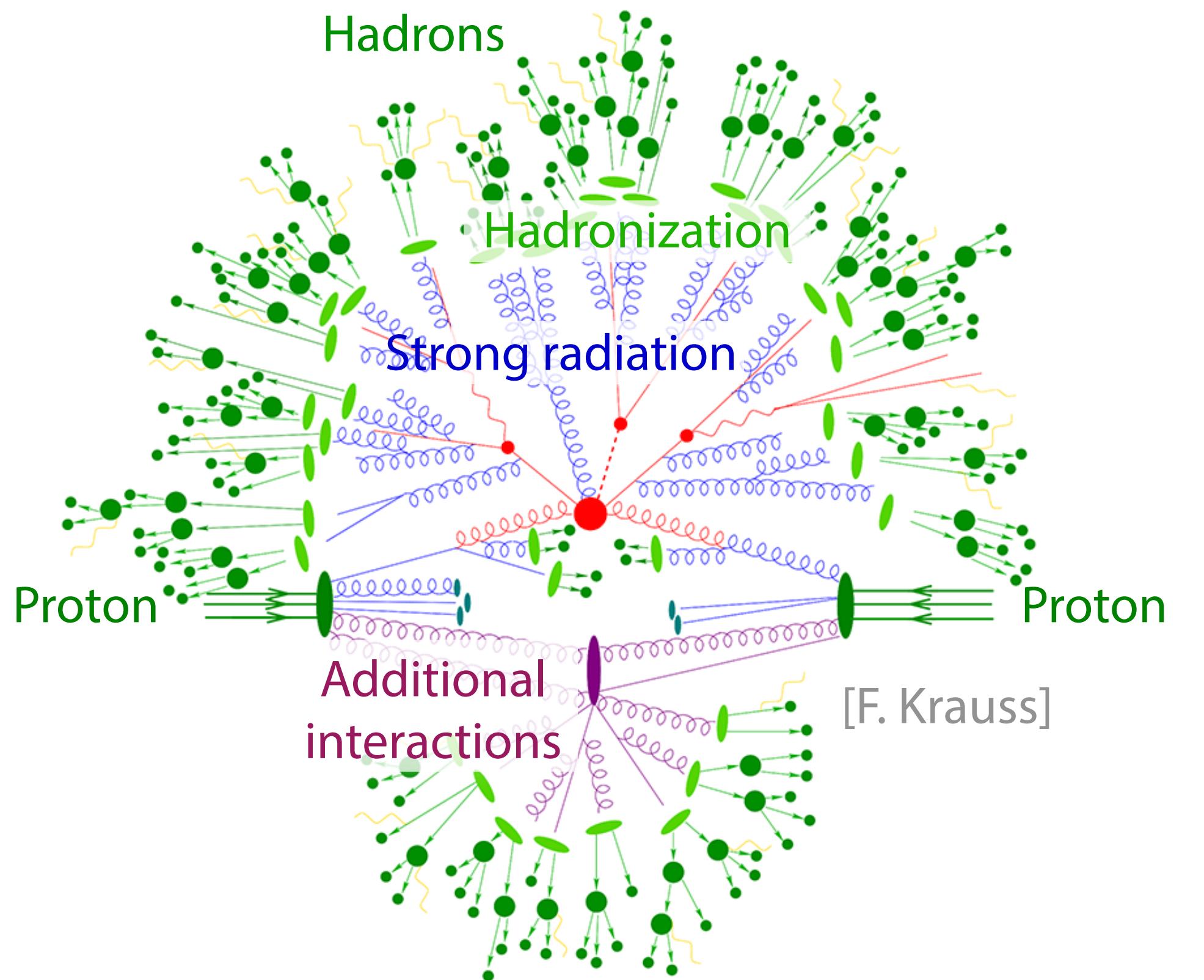
High-energy interactions:

- Rigorously described by quantum field theories
- Sensitive to new physics

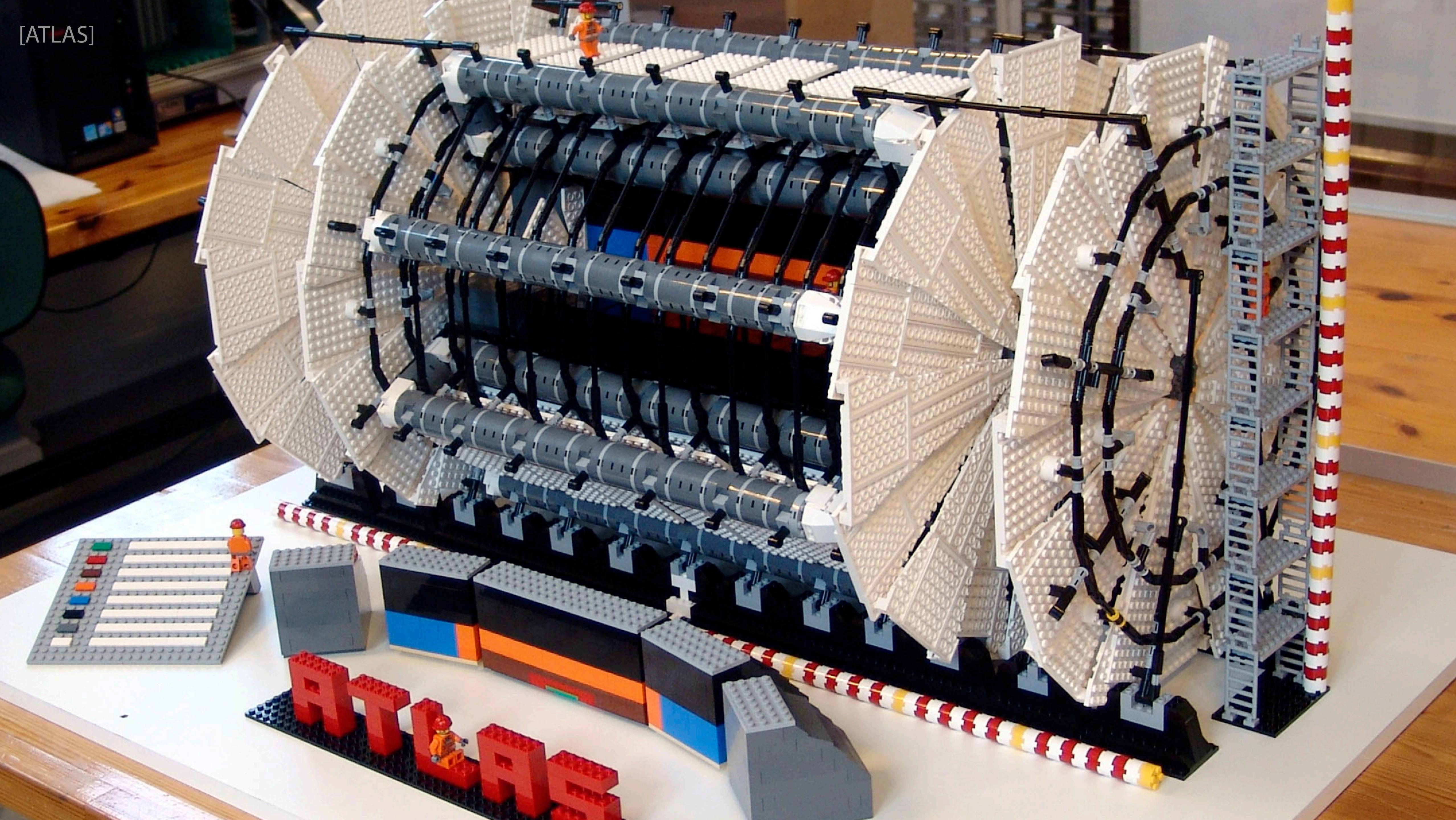


$$\begin{aligned} S = \int d^4x \left[& \mathcal{L}_{\text{SM}} + \frac{f_{\phi,2}}{\Lambda^2} \frac{1}{2} \partial_\mu (\phi^\dagger \phi) \partial^\mu (\phi^\dagger \phi) + \frac{f_{\phi,3}}{\Lambda^2} \frac{1}{3} (\phi^\dagger \phi)^3 \right. \\ & + \frac{f_{GG}}{\Lambda^2} (\phi^\dagger \phi) G_{\mu\nu}^a G^{\mu\nu a} - \frac{f_{BB}}{\Lambda^2} \frac{g'^2}{4} (\phi^\dagger \phi) B_{\mu\nu} B^{\mu\nu} - \frac{f_{WW}}{\Lambda^2} \frac{g^2}{4} (\phi^\dagger \phi) W_{\mu\nu}^a W^{\mu\nu a} \\ & + \frac{f_B}{\Lambda^2} \frac{ig'}{2} (D^\mu \phi)^\dagger D^\nu \phi B_{\mu\nu} + \frac{f_W}{\Lambda^2} \frac{ig}{2} (D^\mu \phi)^\dagger \sigma^a D^\nu \phi W_{\mu\nu}^a \\ & + \frac{f_\ell}{\Lambda^2} (\phi^\dagger \phi) \bar{L}_L \phi \ell_R + \frac{f_u}{\Lambda^2} (\phi^\dagger \phi) \bar{Q}_L \tilde{\phi} u_R + \frac{f_d}{\Lambda^2} (\phi^\dagger \phi) \bar{Q}_L \phi d_R \\ & \left. + \frac{f_{G\widetilde{G}}}{\Lambda^2} (\phi^\dagger \phi) G_{\mu\nu}^a \widetilde{G}^{\mu\nu a} - \frac{f_{B\widetilde{B}}}{\Lambda^2} \frac{g'^2}{4} (\phi^\dagger \phi) B_{\mu\nu} \widetilde{B}^{\mu\nu} - \frac{f_{W\widetilde{W}}}{\Lambda^2} \frac{g^2}{4} (\phi^\dagger \phi) W_{\mu\nu}^a \widetilde{W}^{\mu\nu a} \right] \end{aligned}$$

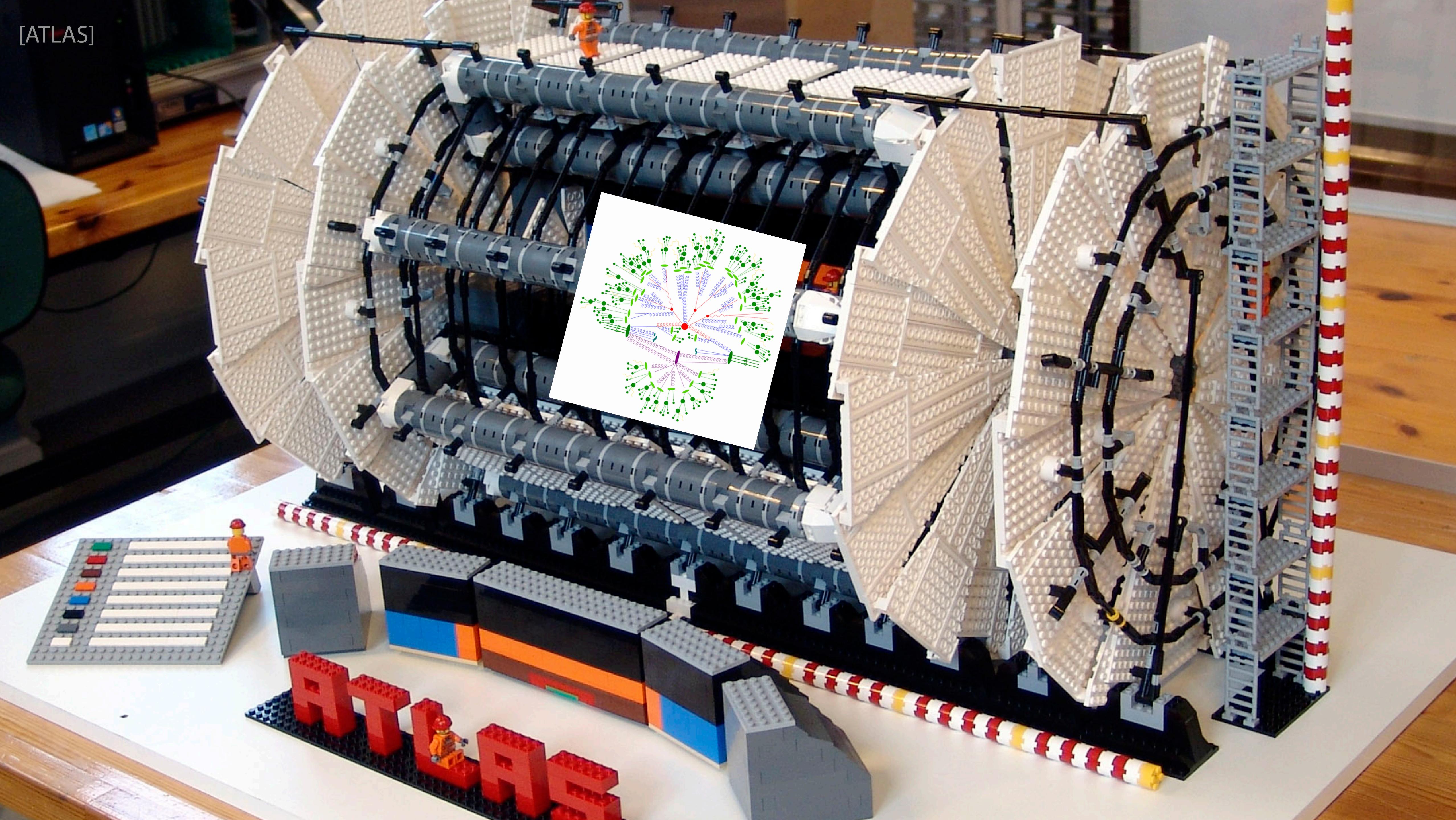
Proton collisions

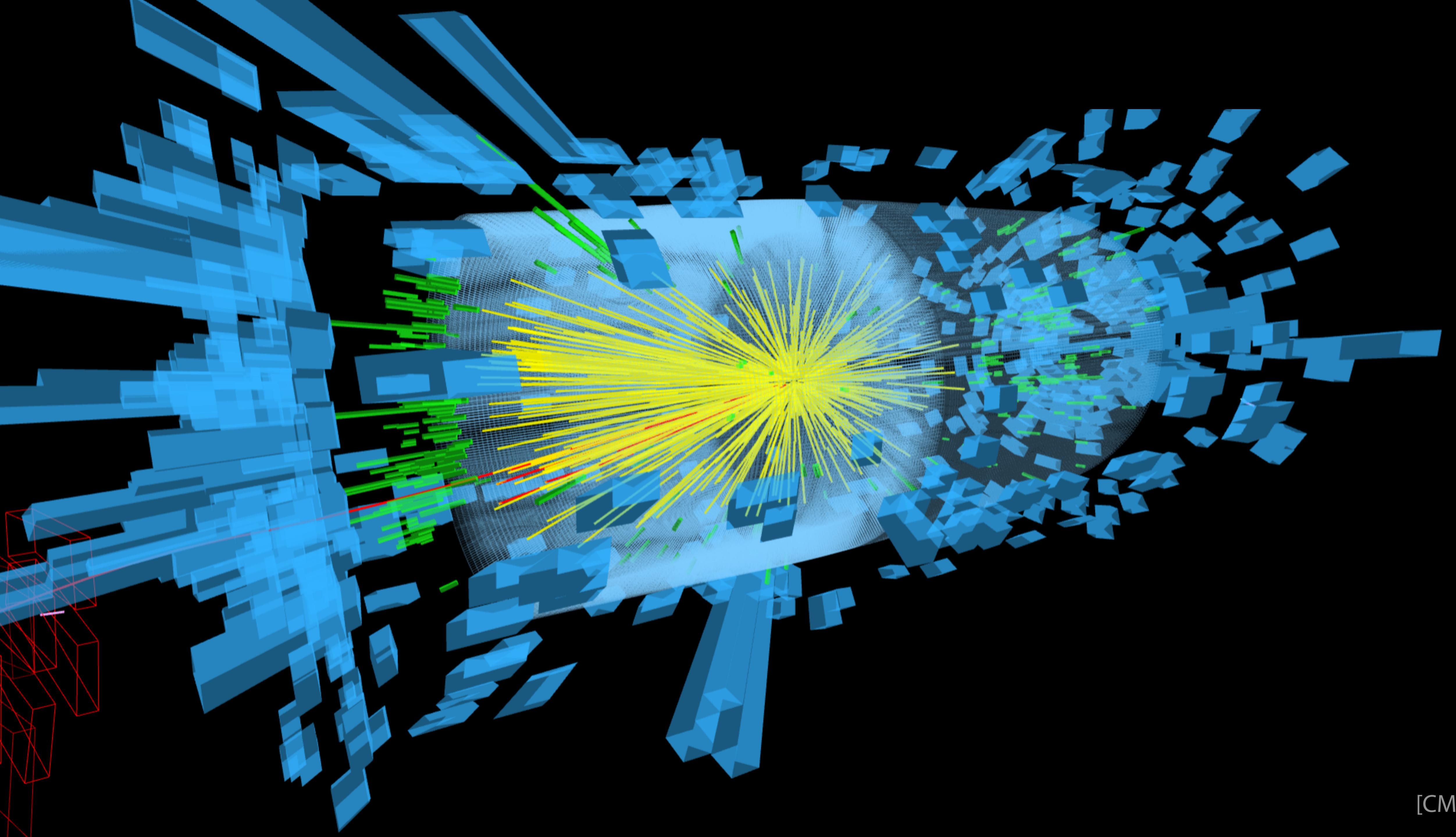


[ATLAS]



[ATLAS]



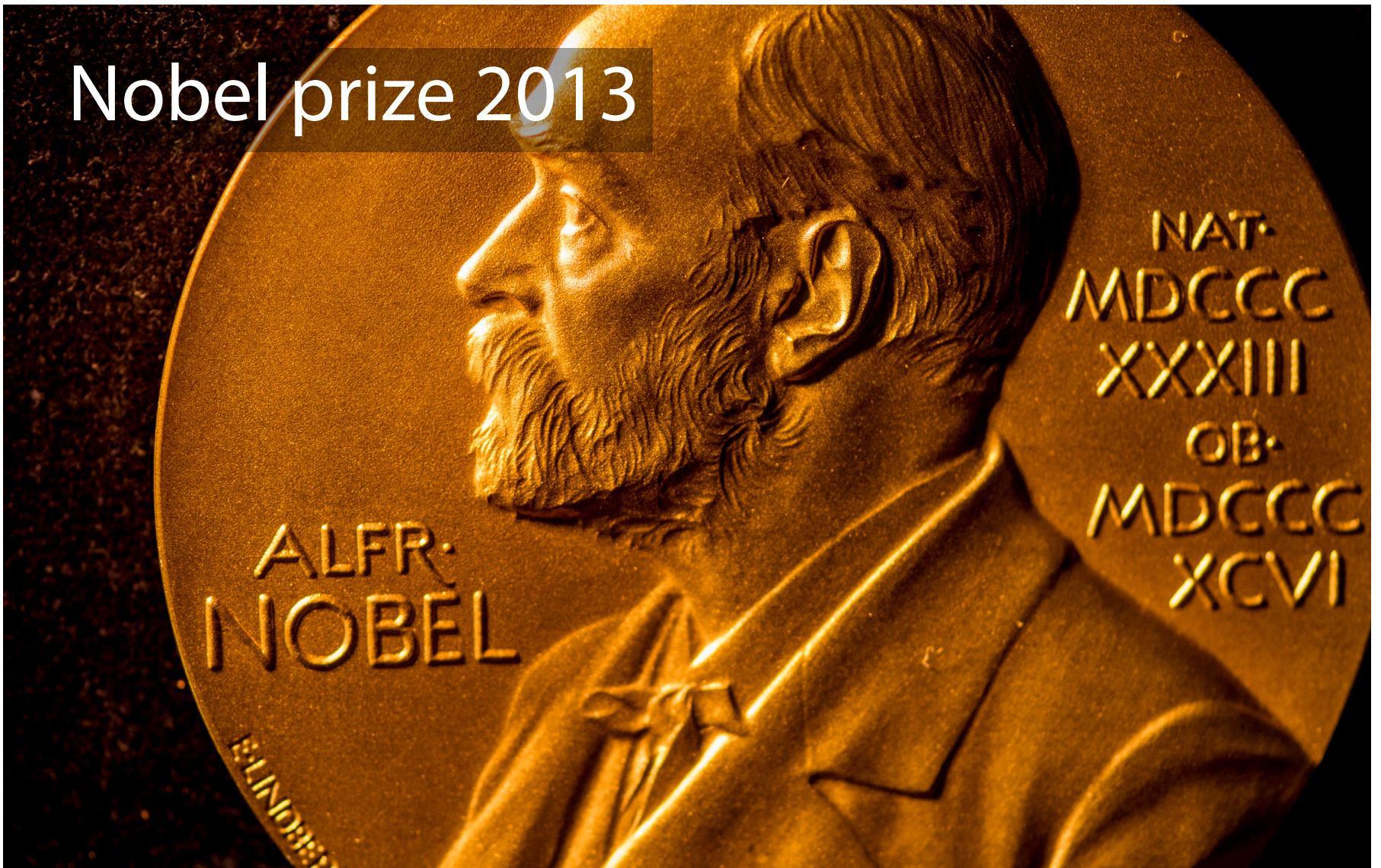


[CMS]

Understanding the Higgs boson



Discovery 2012

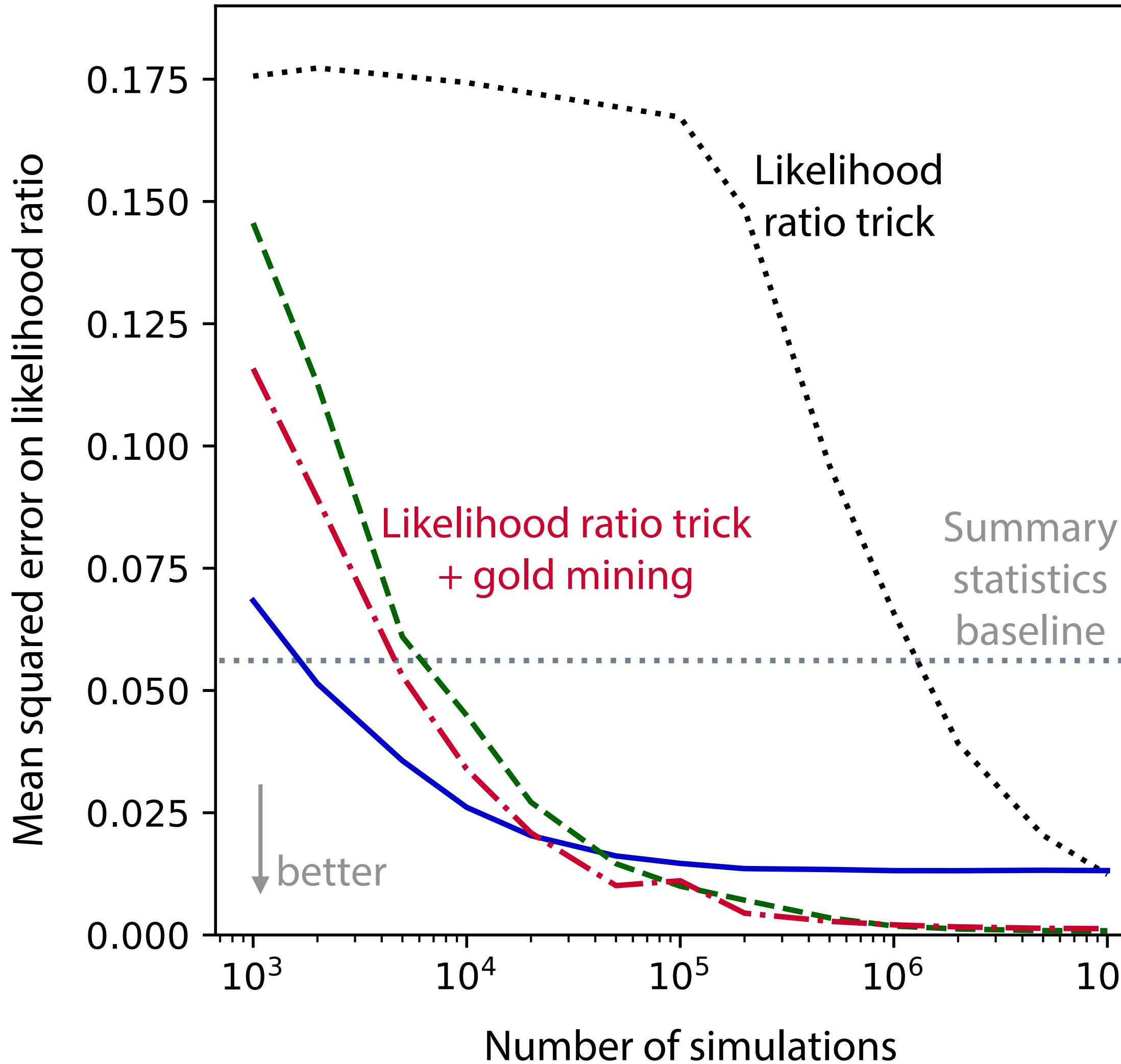


Nobel prize 2013

Now, precision measurements of Higgs properties and interactions may help us answer big questions:

- Why is gravity so weak?
- Why are some particles so light and others so heavy?
- Why is there more matter than antimatter?
- Is the vacuum stable?

Improving sample efficiency

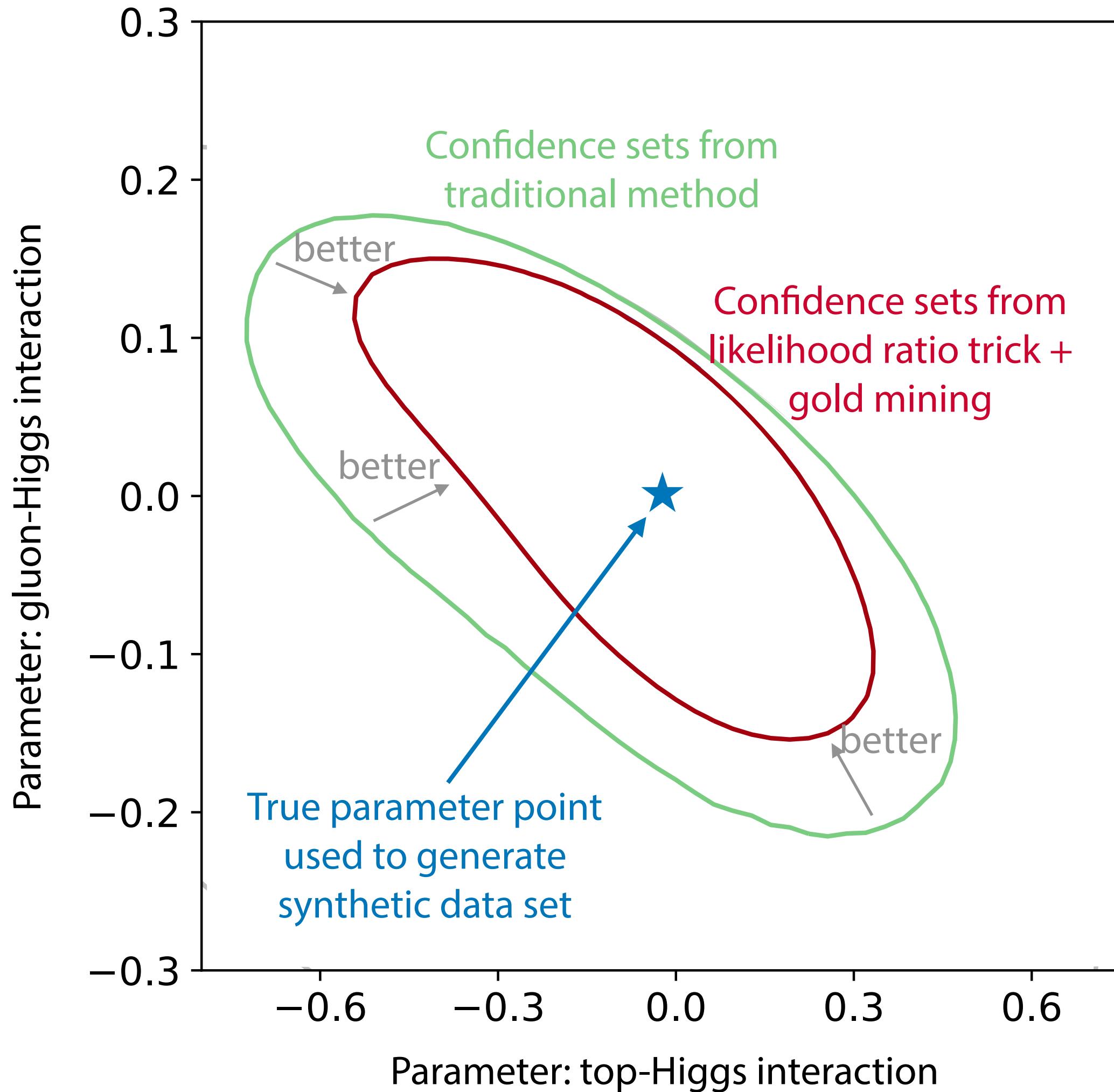


With enough training data, the ML algorithms get the likelihood function right.

Using more information from the simulator improves sample efficiency substantially.

[JB, K. Cranmer, G. Louppe, J. Pavez 1805.00013; 1805.00020;
M. Stoye, JB, K. Cranmer, G. Louppe, J. Pavez 1808.00973]

Improving quality of inference



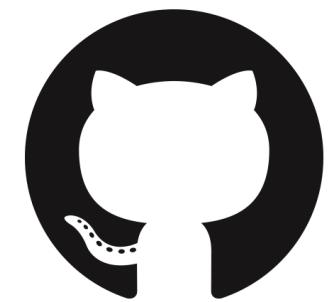
In some processes, the ML-based inference techniques improve the precision as much as taking 90% more data would!

[JB, K. Cranmer, G. Louppe, J. Pavez 1805.00013; 1805.00020;
JB, F. Kling, I. Espejo, K. Cranmer 1907.10621]

Automation

[JB, F. Kling, I. Espejo, K. Cranmer 1907.10621]

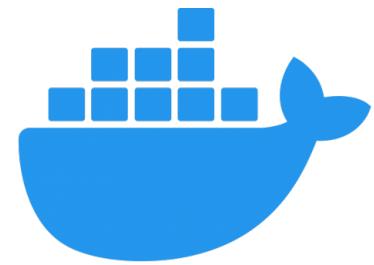
Our open-source Python package **MadMiner** makes it straightforward to apply these ML-based inference techniques



github.com/diana-hep/madminer



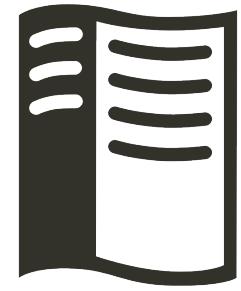
`pip install madminer`



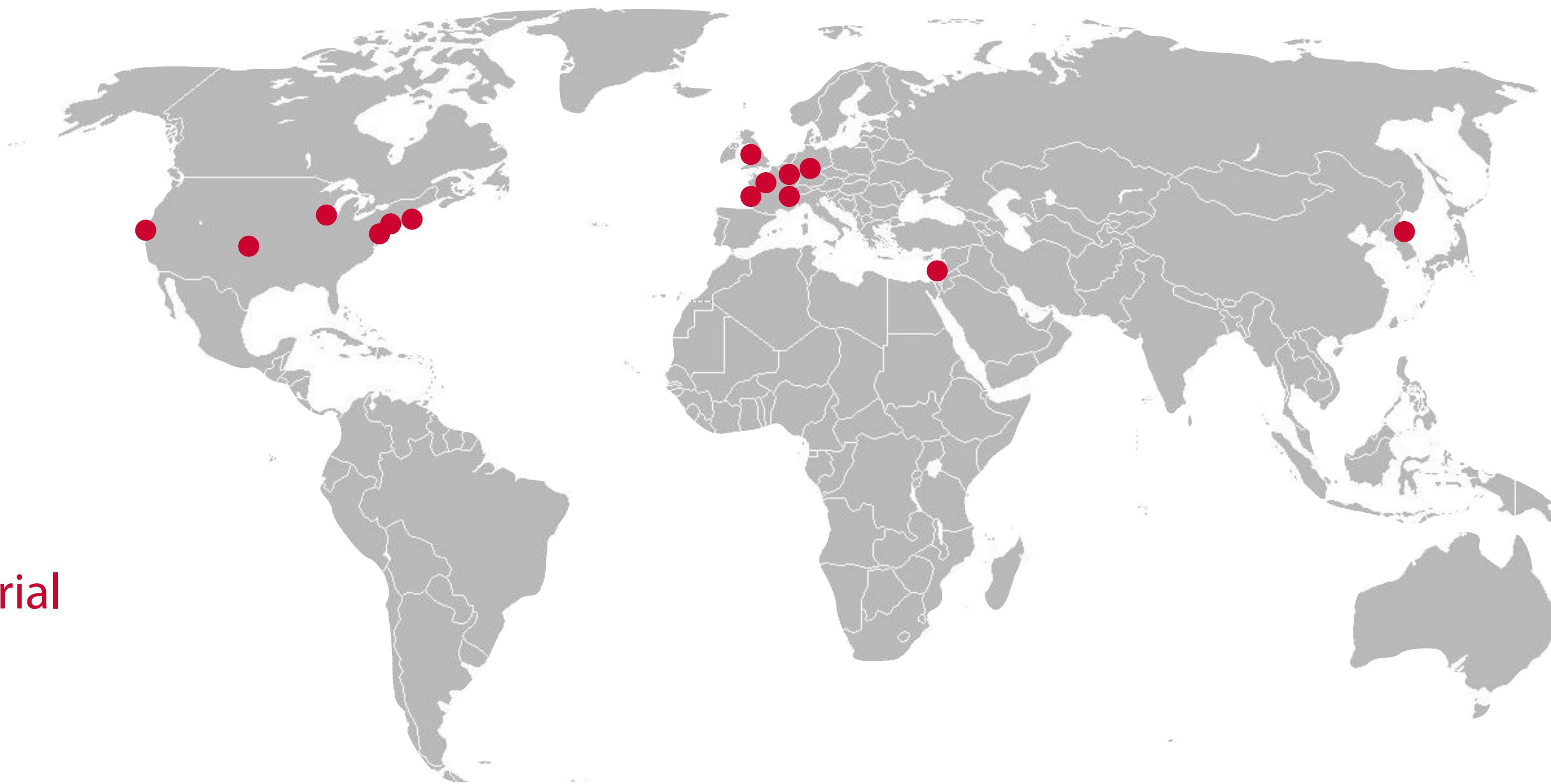
hub.docker.com/u/madminertool



cranmer.github.io/madminer-tutorial



madminer.readthedocs.io



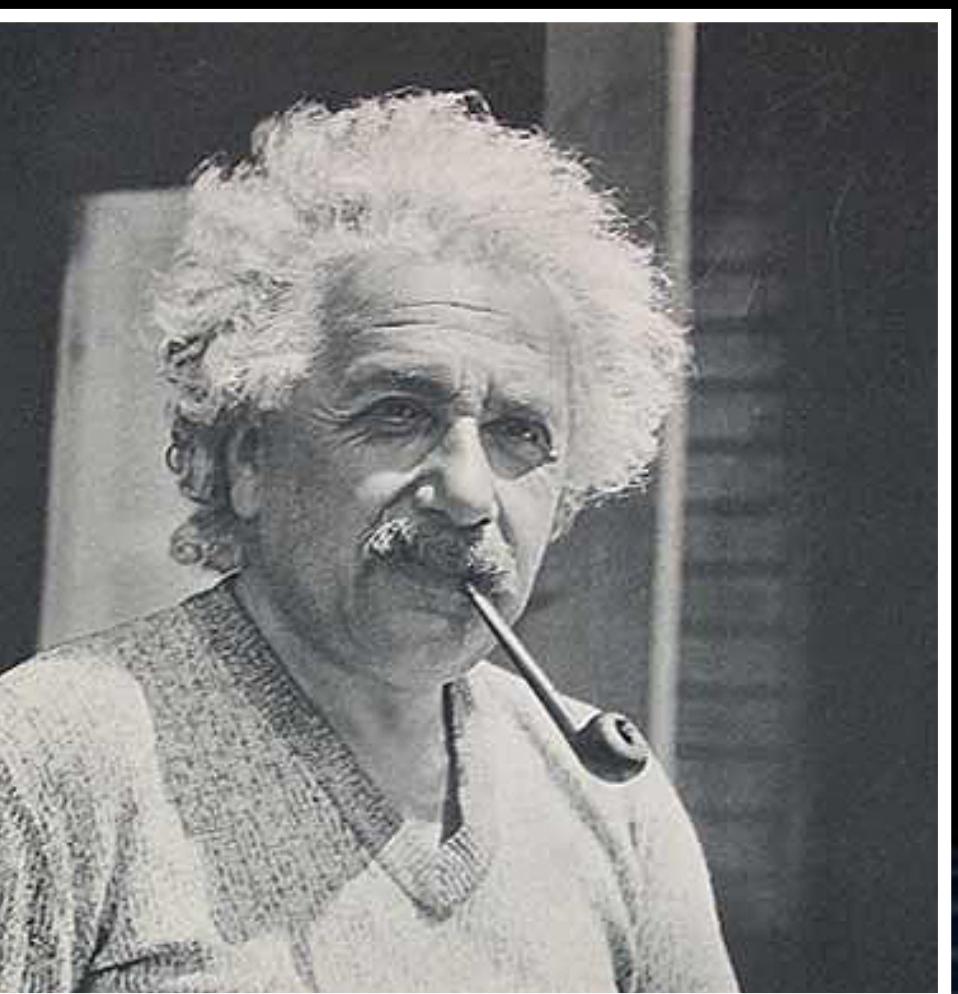
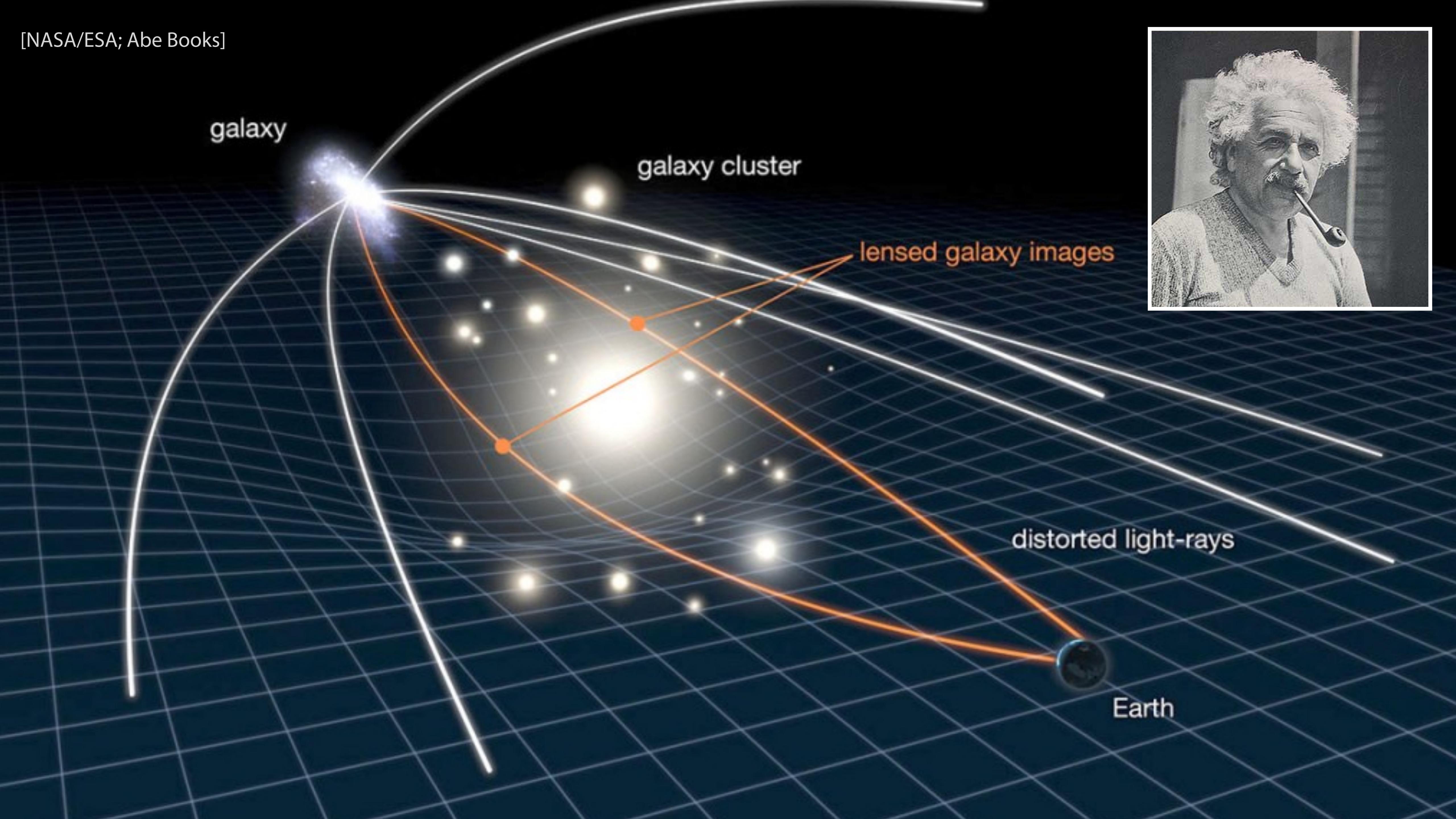


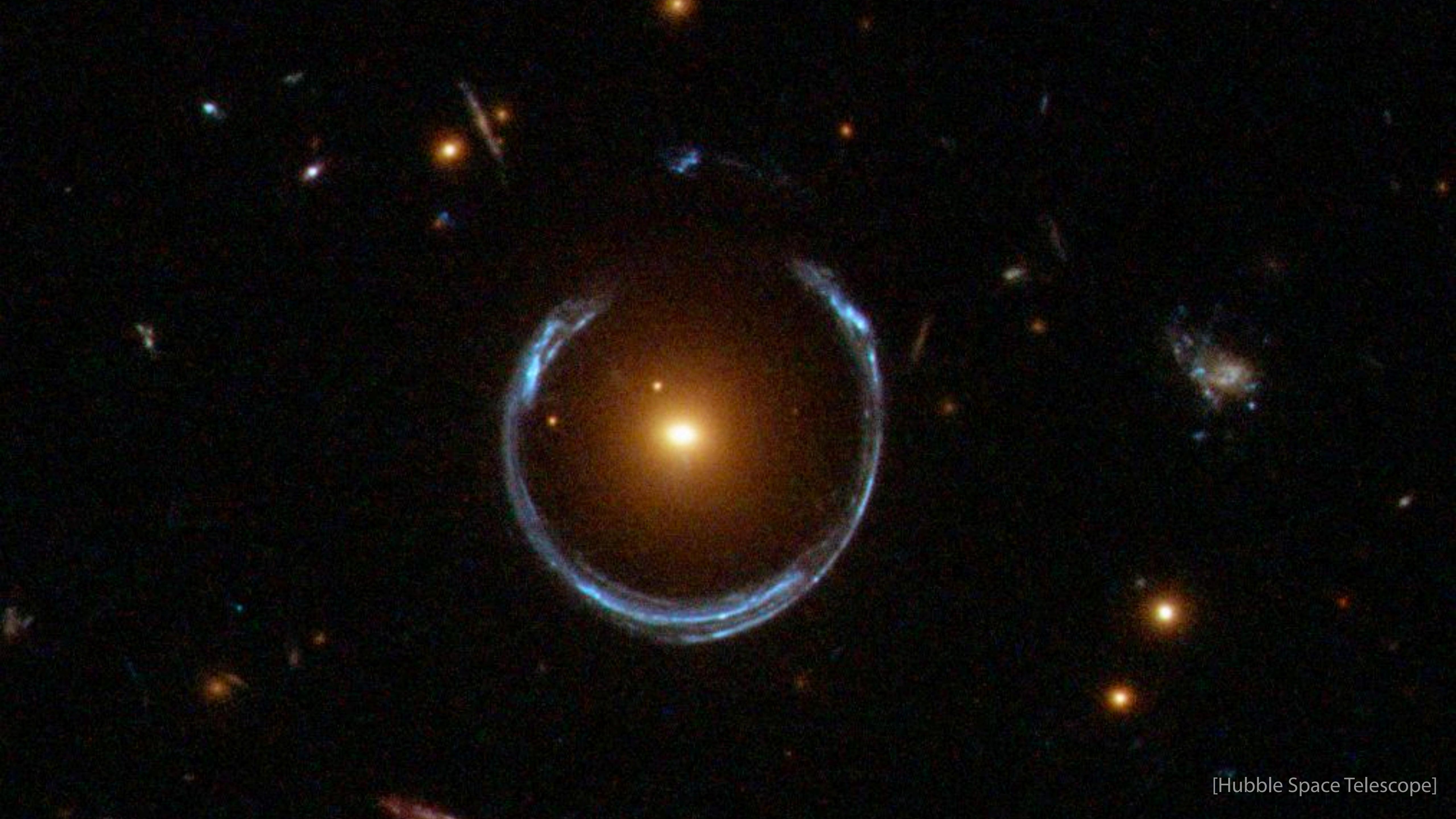
4. Astrophysics

[T. Brown, J.Tumlinson]



[NASA/ESA; Abe Books]

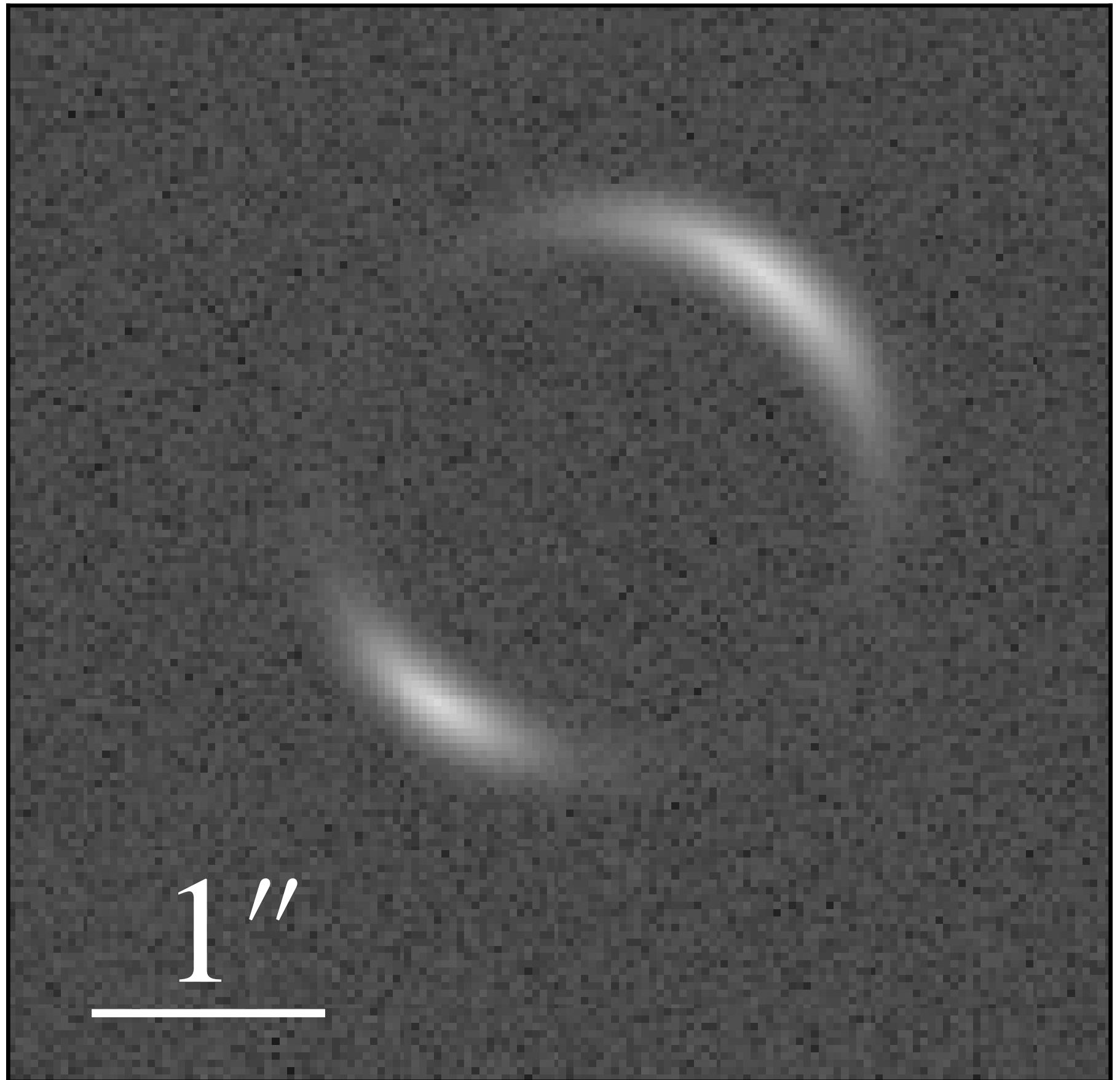




[Hubble Space Telescope]

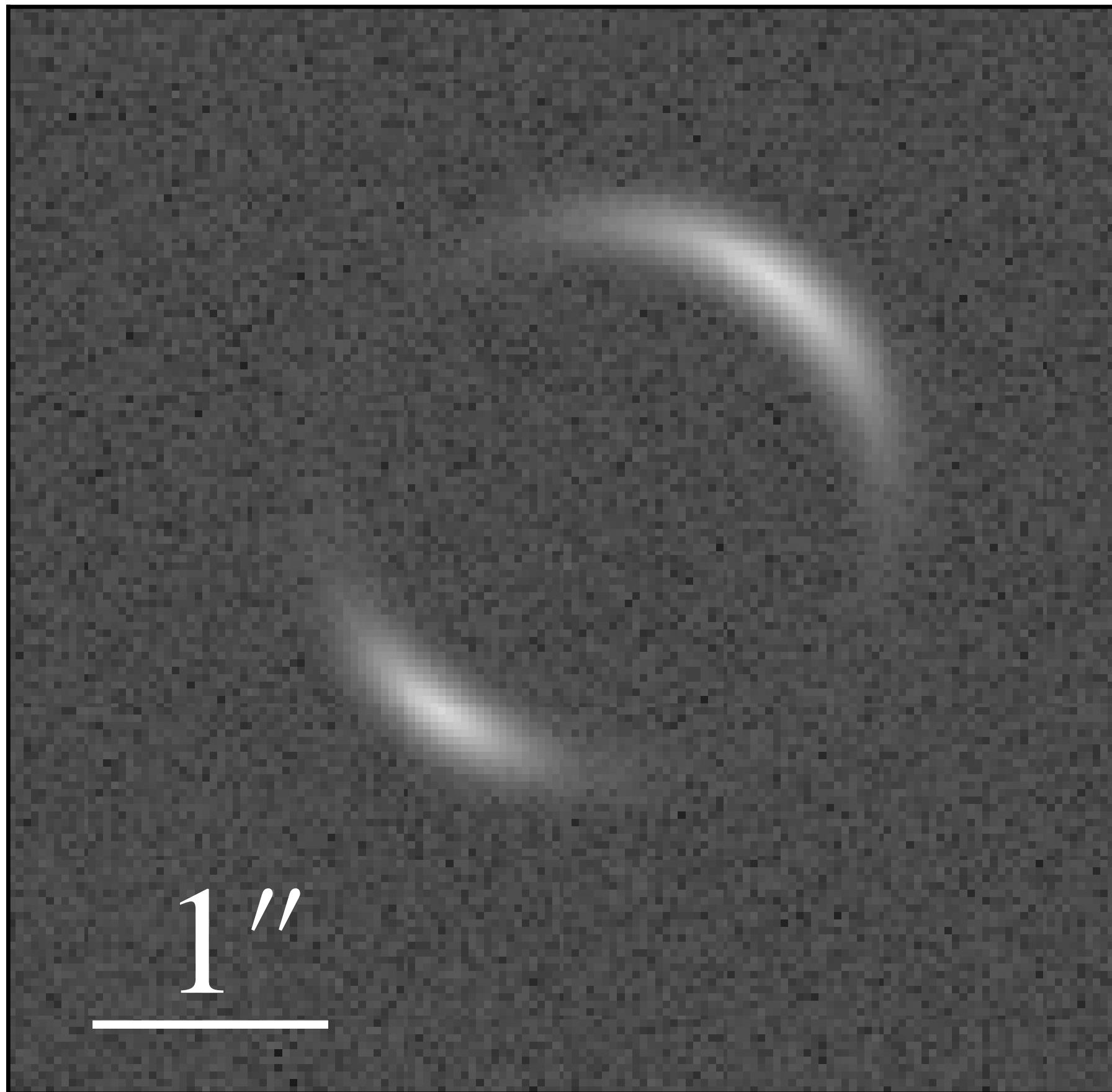
Subhalos affect strong lensing

Smooth halo only

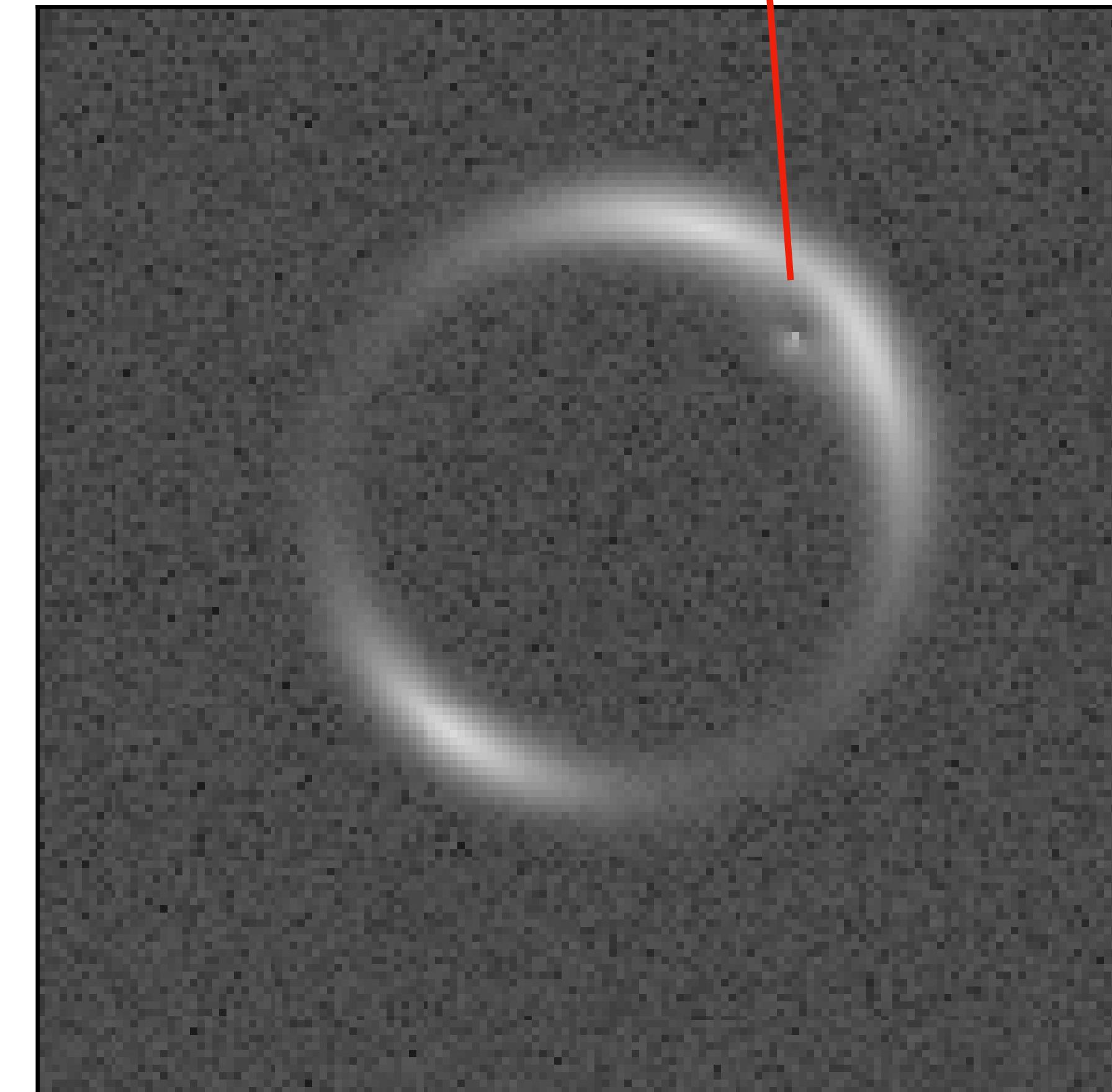


Subhalos affect strong lensing

Smooth halo only

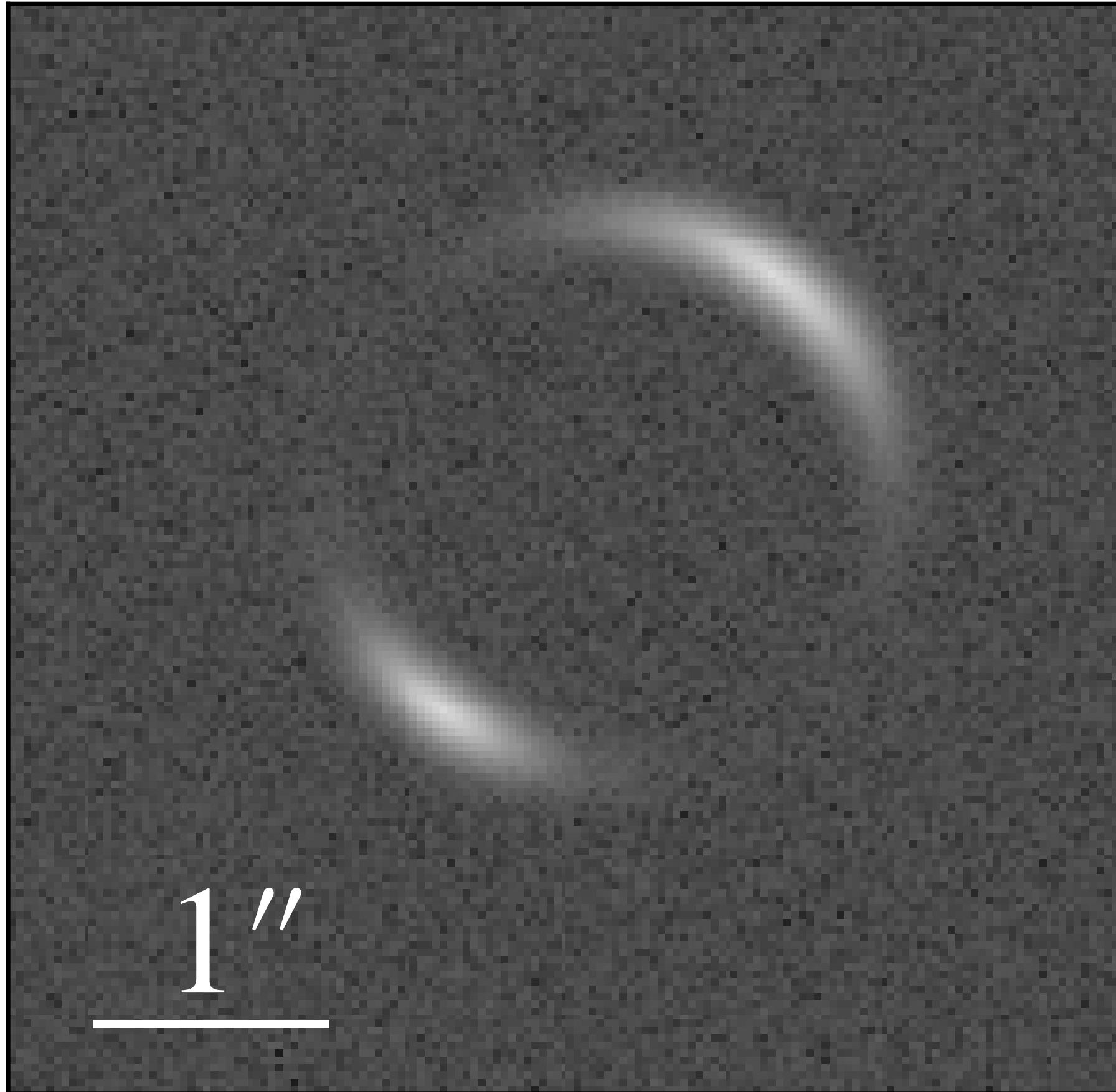


Smooth halo + **subhalo**

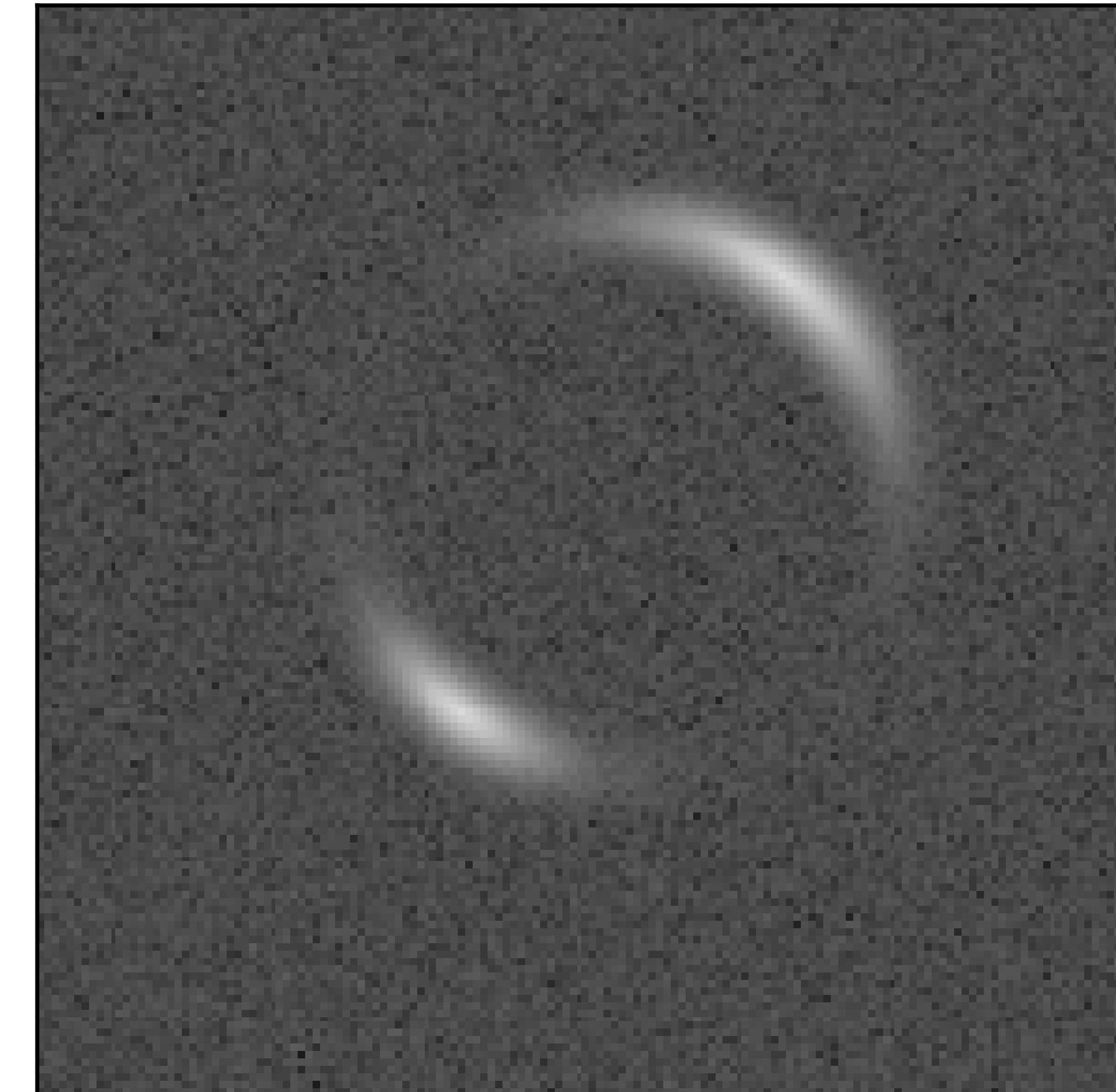


Subhalos affect strong lensing... realistically

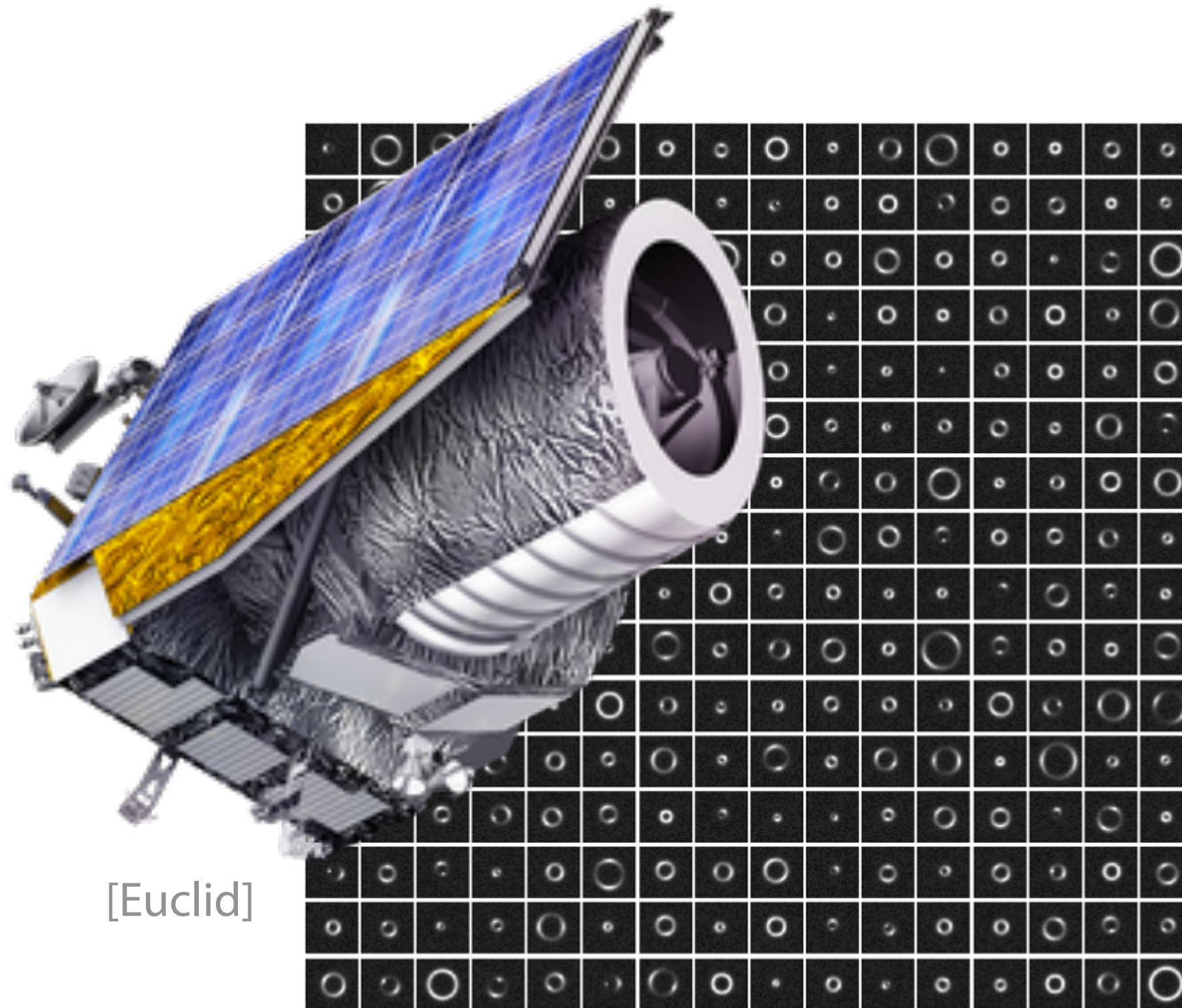
Smooth halo only



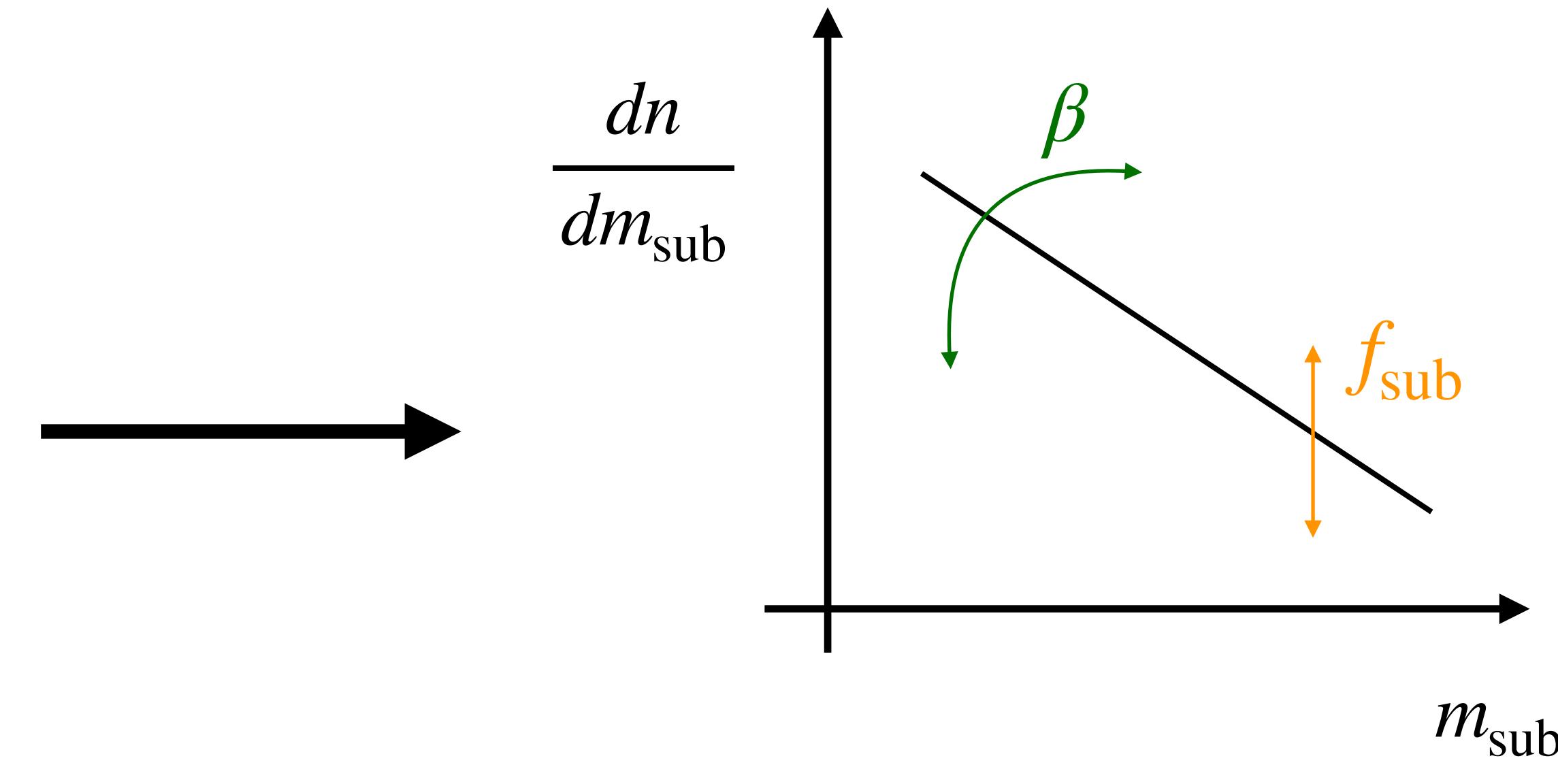
Smooth halo + subhalos



Scalable inference for small subhalos



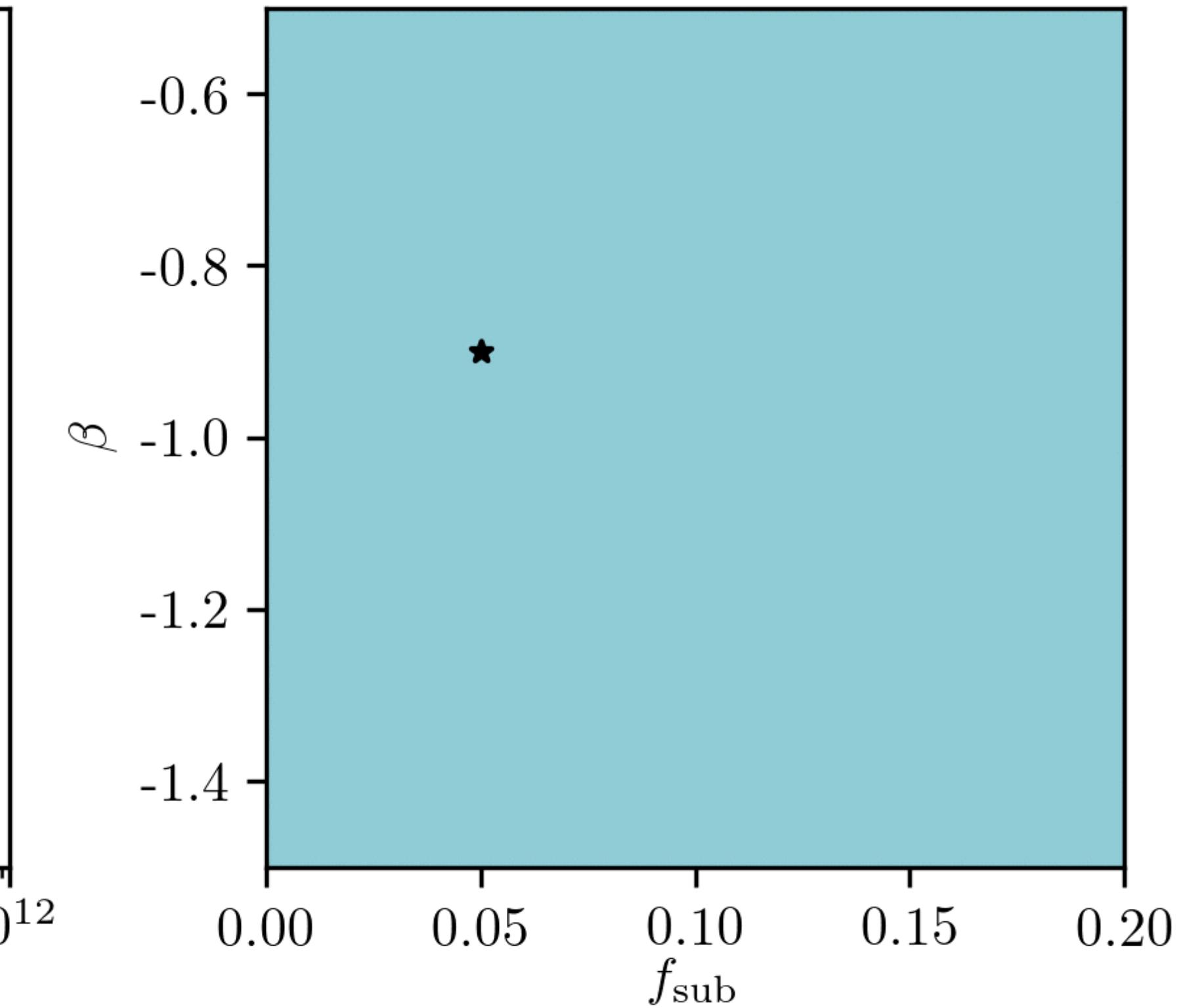
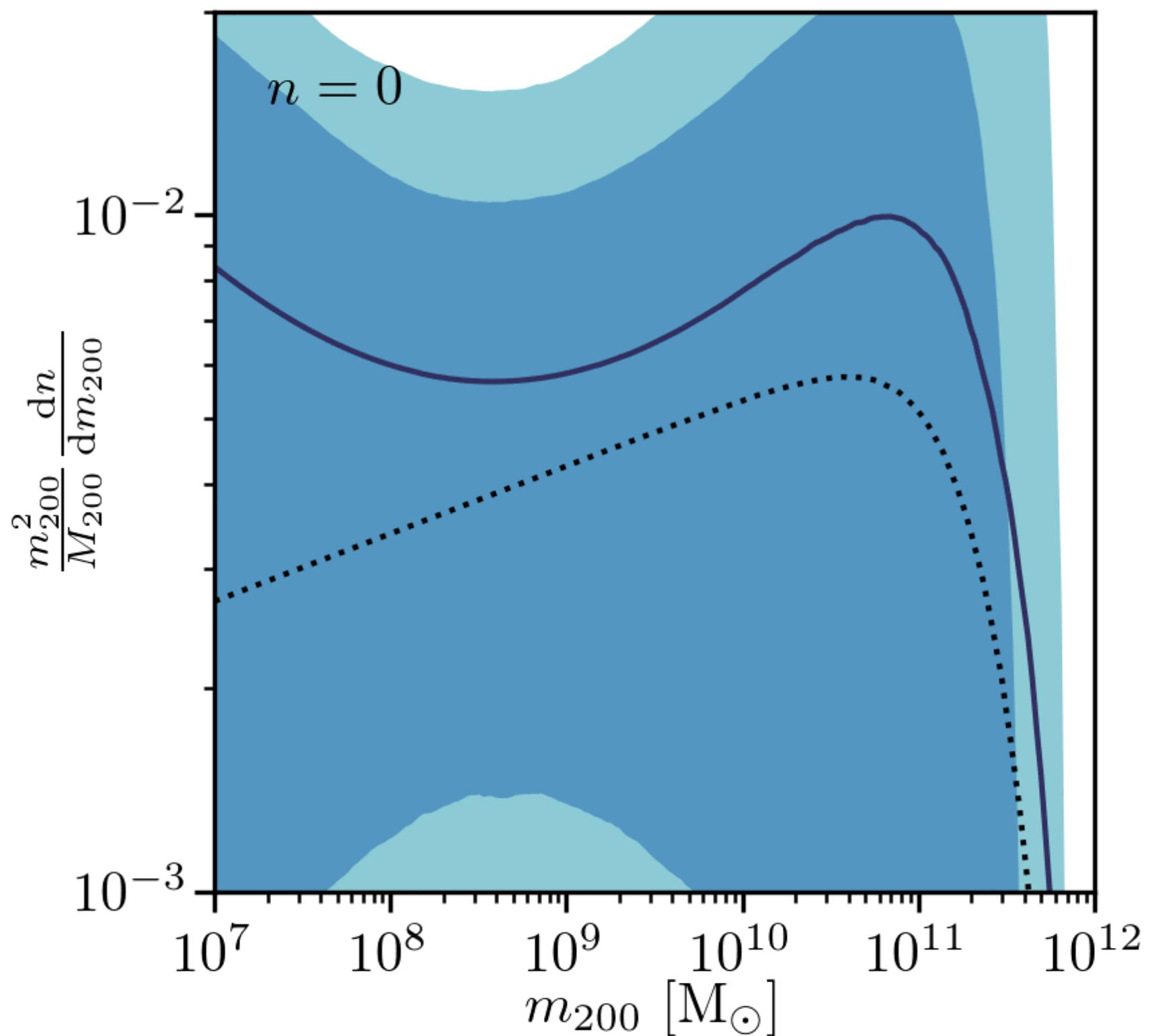
Near-future telescopes and satellites will collect hundreds of lensing images [Collett et al 1507.02657]



Goal: infer DM properties from all images and all clumps at once

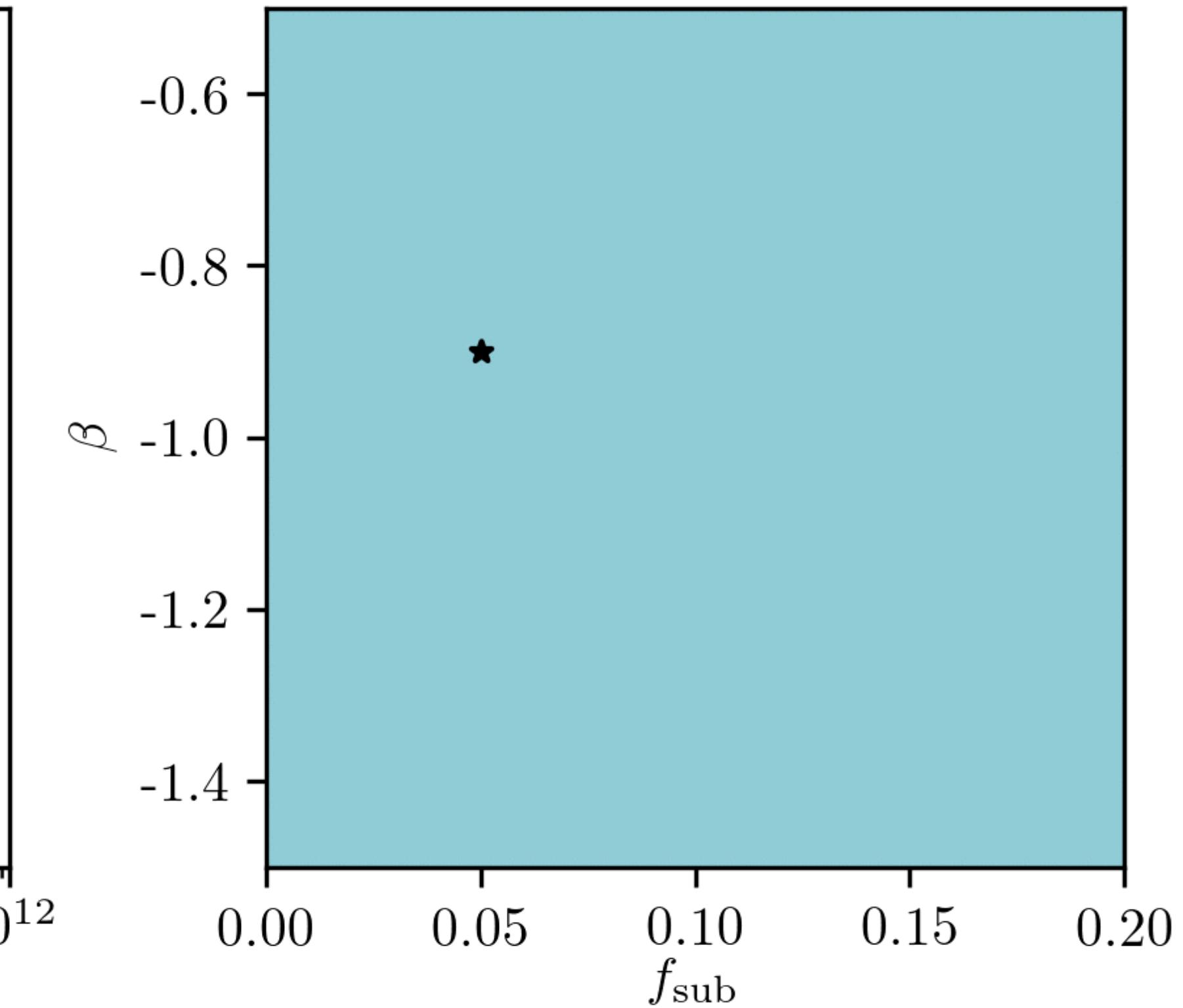
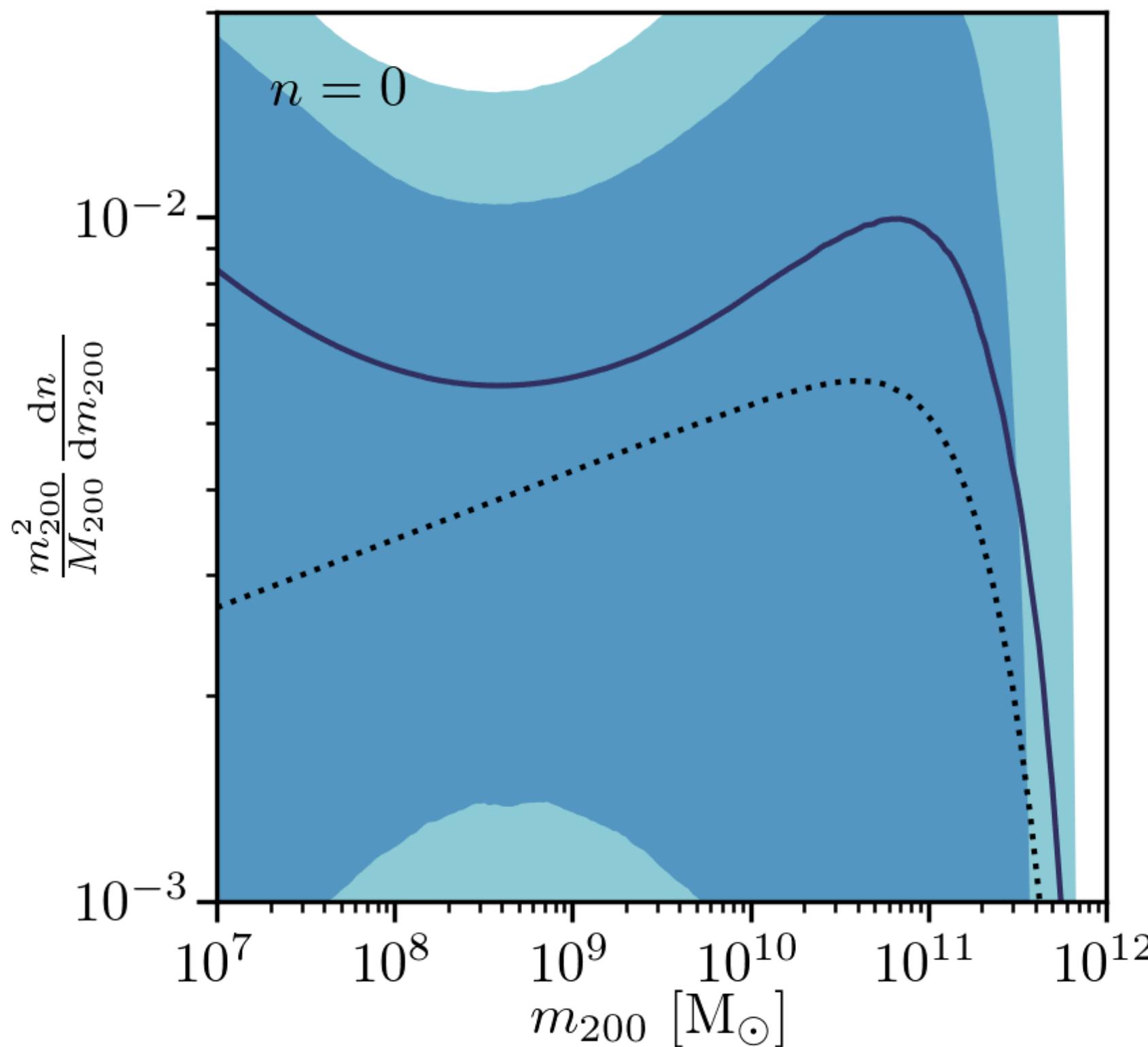
ML-based Bayesian inference

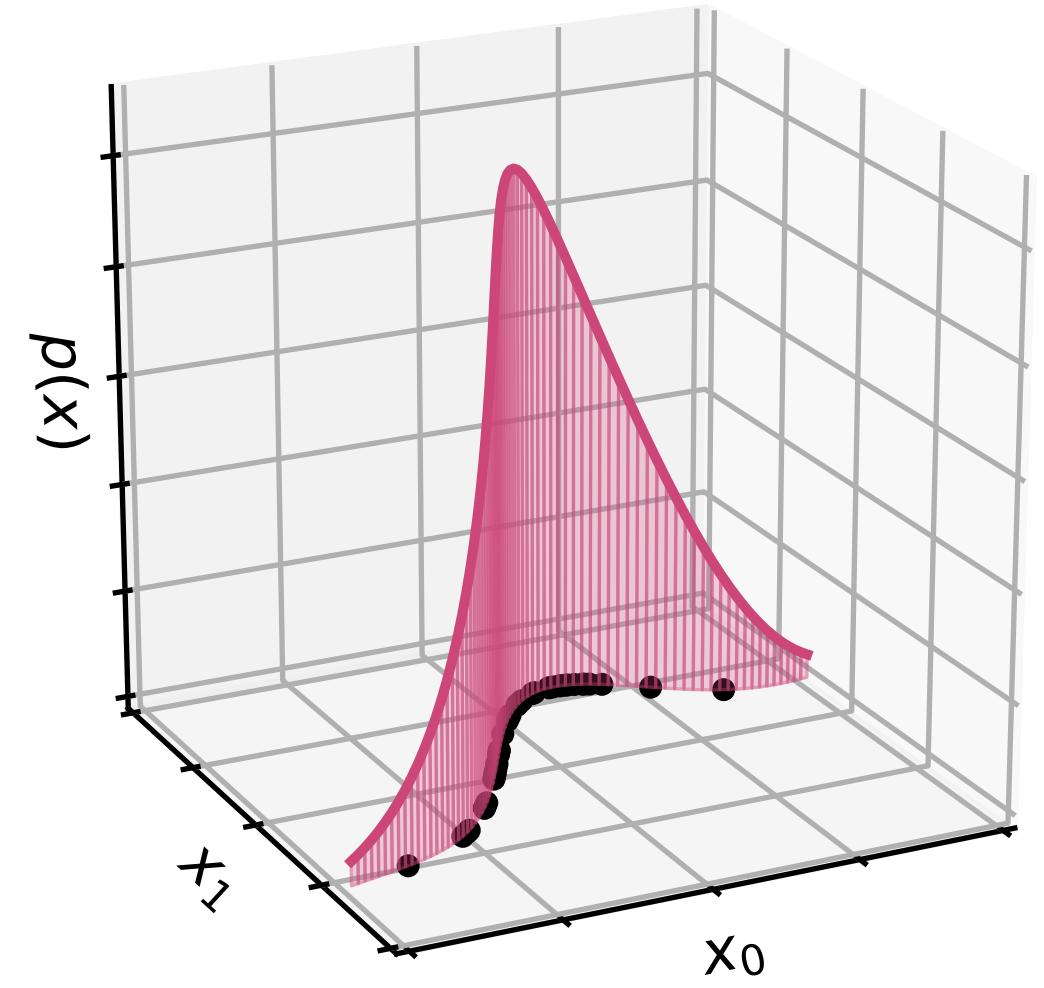
[JB, S. Mishra-Sharma, J. Hermans, G. Louuppe, K. Cranmer 1909.02005]



ML-based Bayesian inference

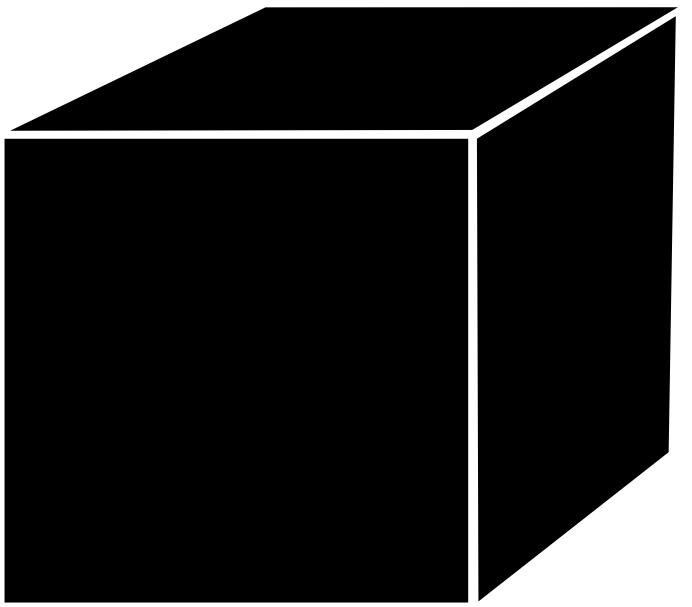
[JB, S. Mishra-Sharma, J. Hermans, G. Louuppe, K. Cranmer 1909.02005]





5. Tangents

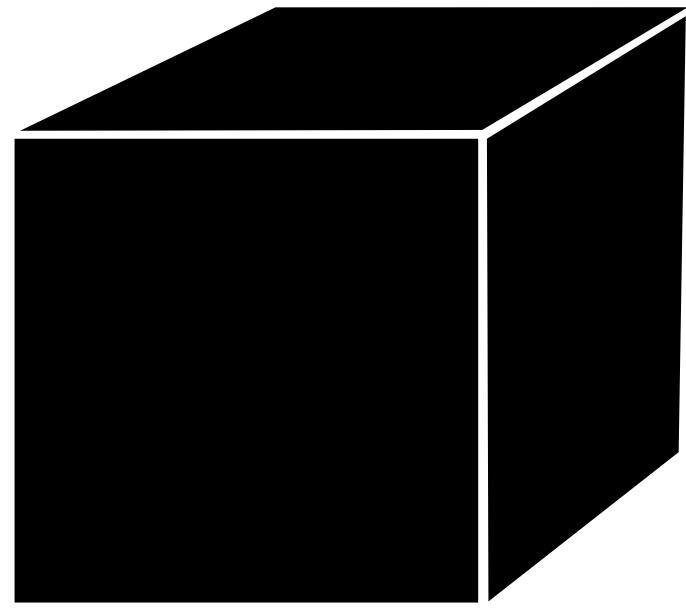
The Achilles heel: model misspecification



Can you trust the simulator?

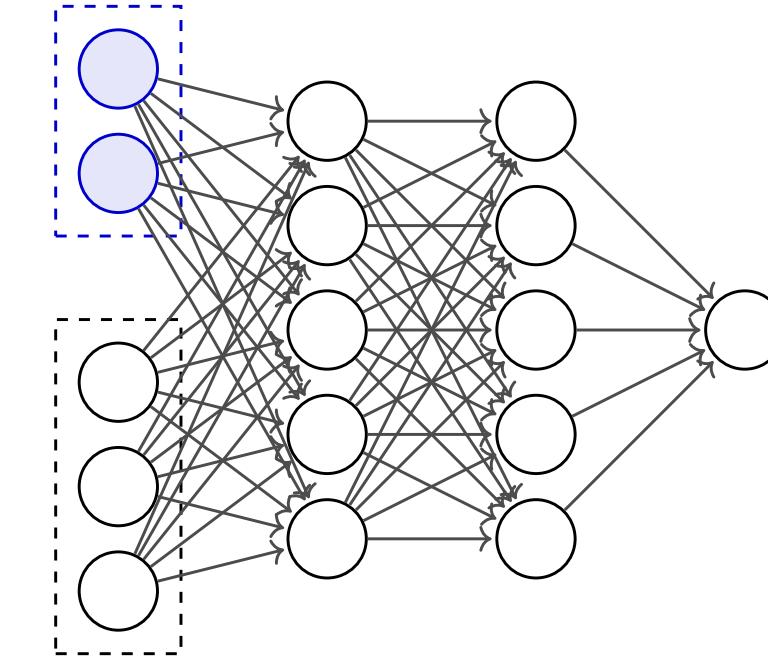
- Model uncertainties explicitly:
nuisance parameters + profiling / marginalization
- Make analysis robust:
ideas from domain adaptation, algorithmic fairness
[G. Louppe, M. Kagan, K. Cranmer 1611.01046; J. Alsing, B. Wandelt 1903.01473; P. de Castro, T. Dorigo 1806.04743]

The Achilles heel: model misspecification



Can you trust the simulator?

- Model uncertainties explicitly:
nuisance parameters + profiling / marginalization
- Make analysis robust:
ideas from domain adaptation, algorithmic fairness
[G. Louppe, M. Kagan, K. Cranmer 1611.01046; J. Alsing, B. Wandelt 1903.01473; P. de Castro, T. Dorigo 1806.04743]



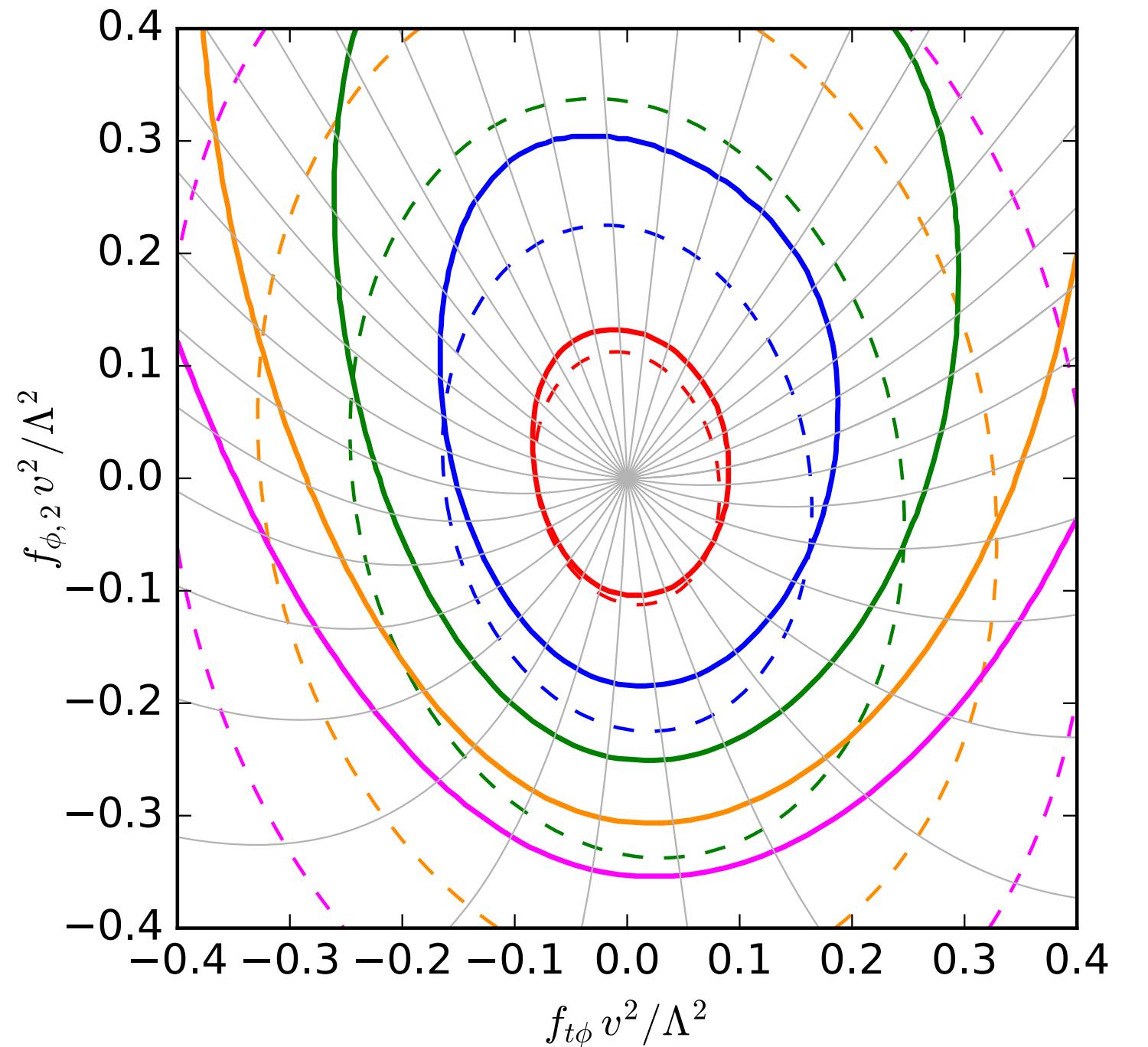
Can you trust the neural network?

- Sanity checks: expectation values, “critic” tests
- Calibration / Neyman construction with toys
(badly trained network can lead to suboptimal limits, but not to wrong limits)
[JB, G. Louppe, J. Pavez, K. Cranmer 1805.00020]

Information geometry

[JB, K. Cranmer, F. Kling, T. Plehn 1612.05261;
JB, F. Kling, T. Plehn, T. Tait 1712.02350]

Study manifold of probability
distributions geometrically

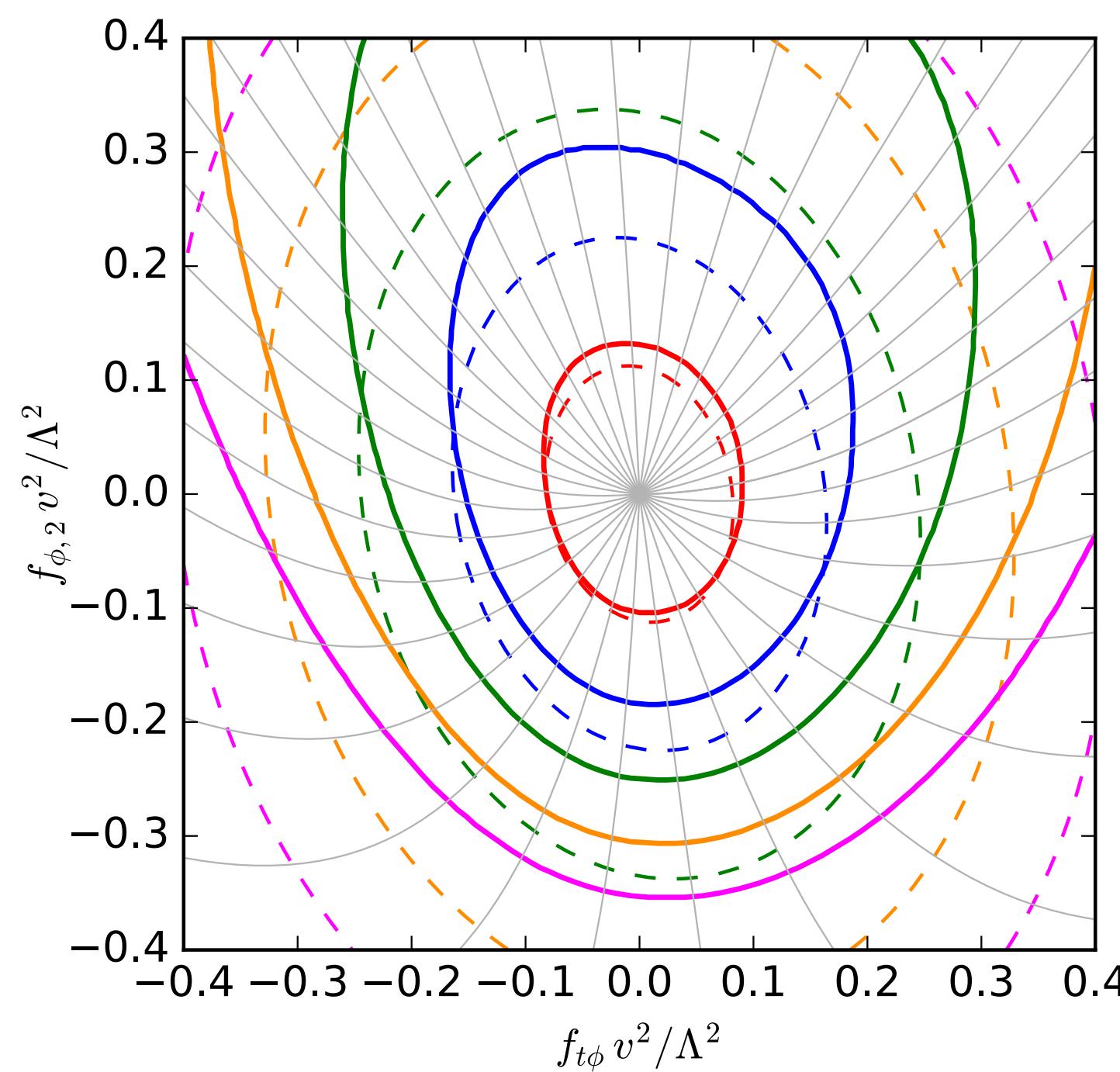


Distances \leftrightarrow sensitivity with which an experiment can distinguish parameter points

Information geometry

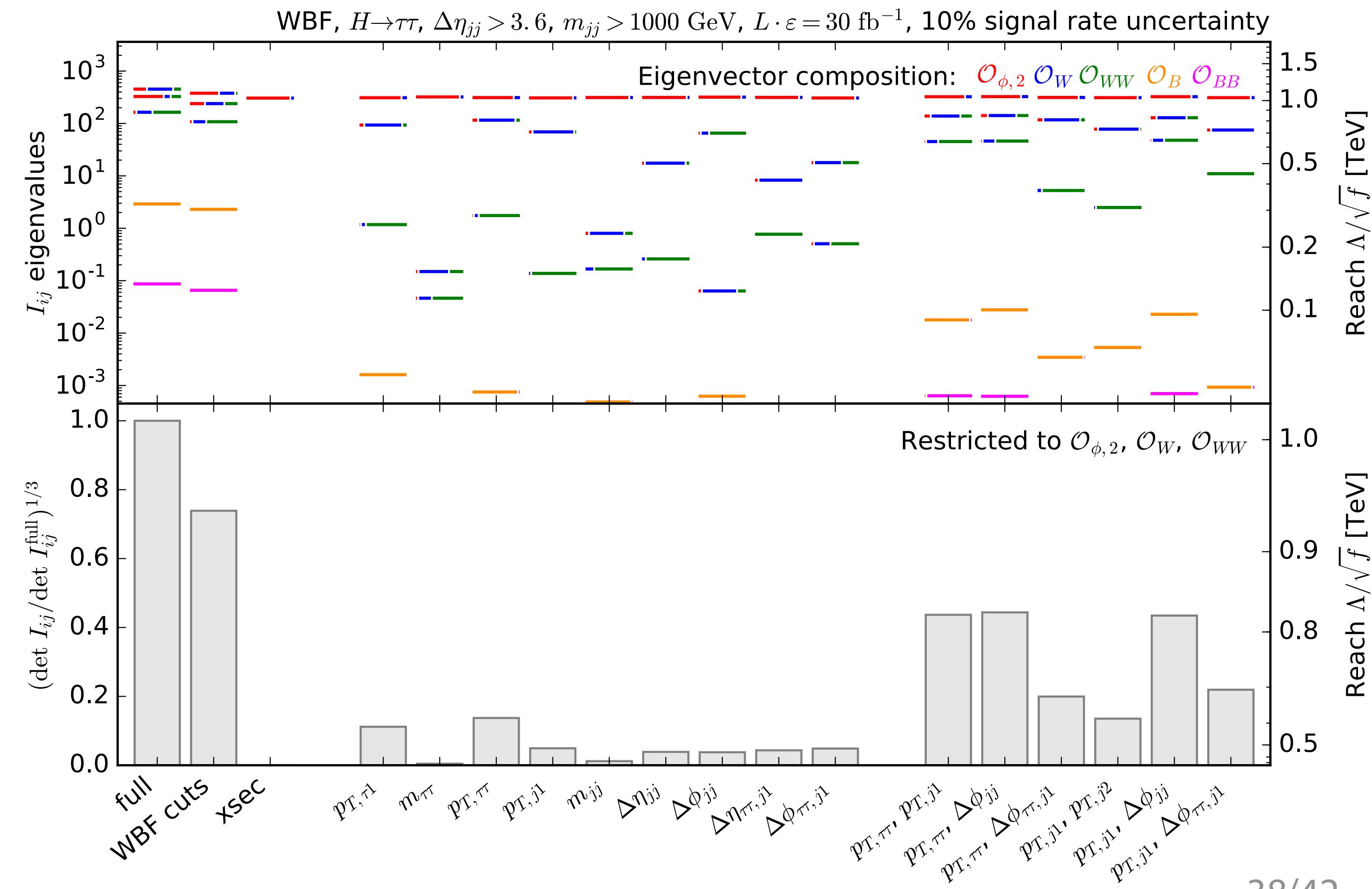
[JB, K. Cranmer, F. Kling, T. Plehn 1612.05261;
JB, F. Kling, T. Plehn, T. Tait 1712.02350]

Study manifold of probability distributions geometrically



Distances \leftrightarrow sensitivity with which an experiment can distinguish parameter points

In practice, we can use this for sensitivity forecasting, feature selection, experimental design



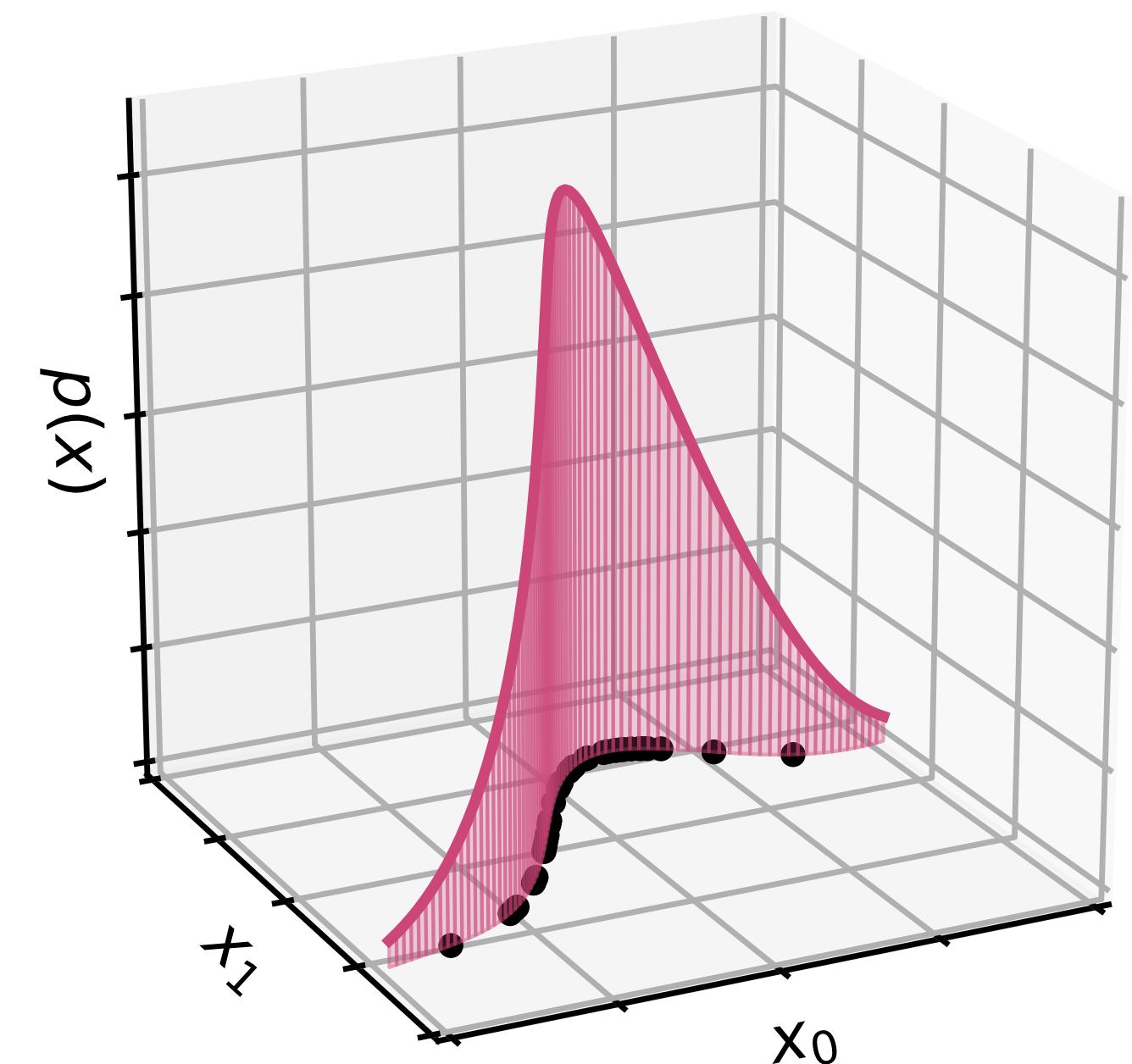
\mathcal{M} -flows

[JB, K. Cranmer 2003.13913]

Often data is restricted to a lower-dimensional manifold embedded in the data space

\mathcal{M} -flows are a new probabilistic / generative model that

- describe data as a tractable probability density on a lower-dimensional manifold
- learn manifold and density from data



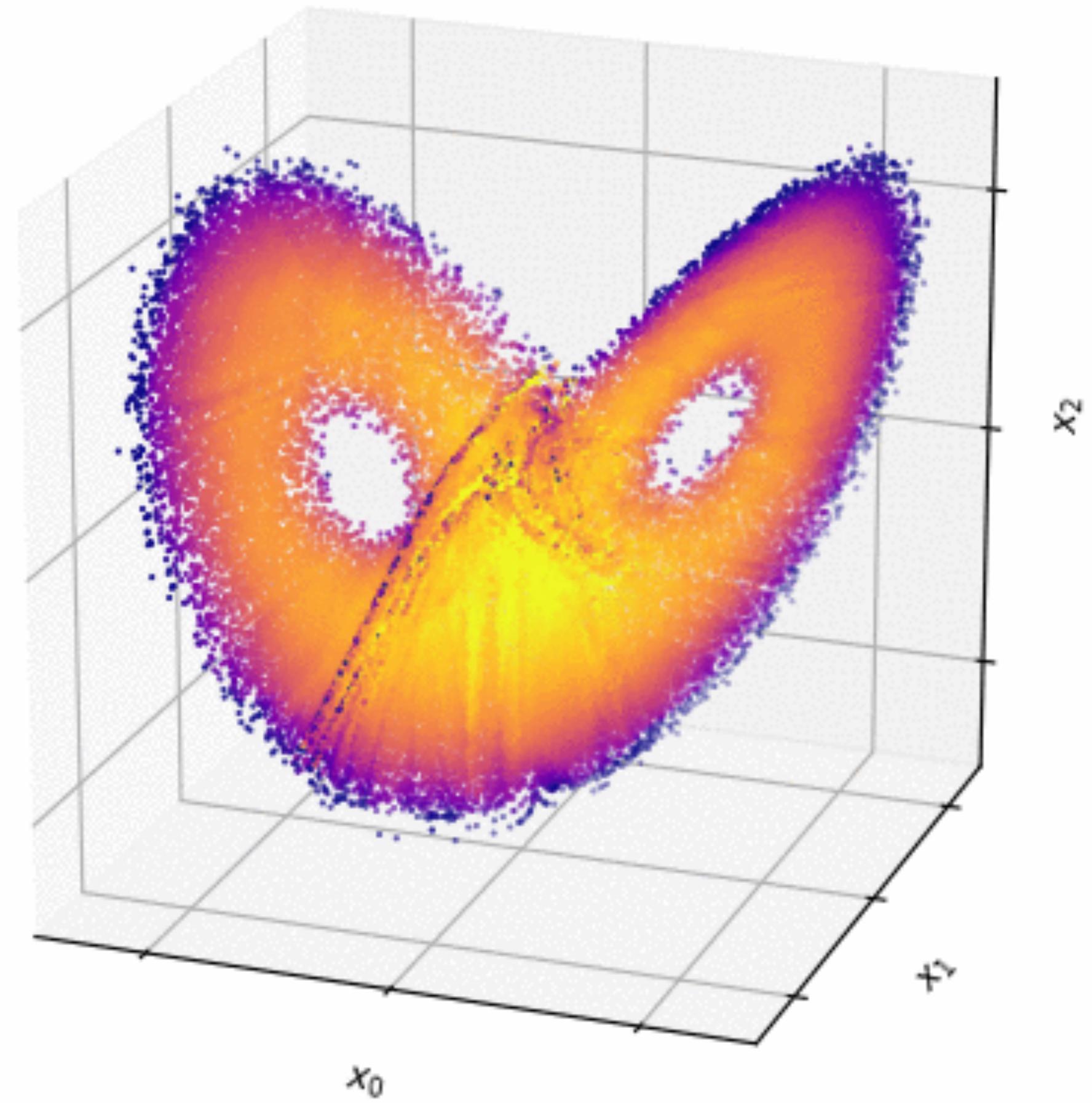
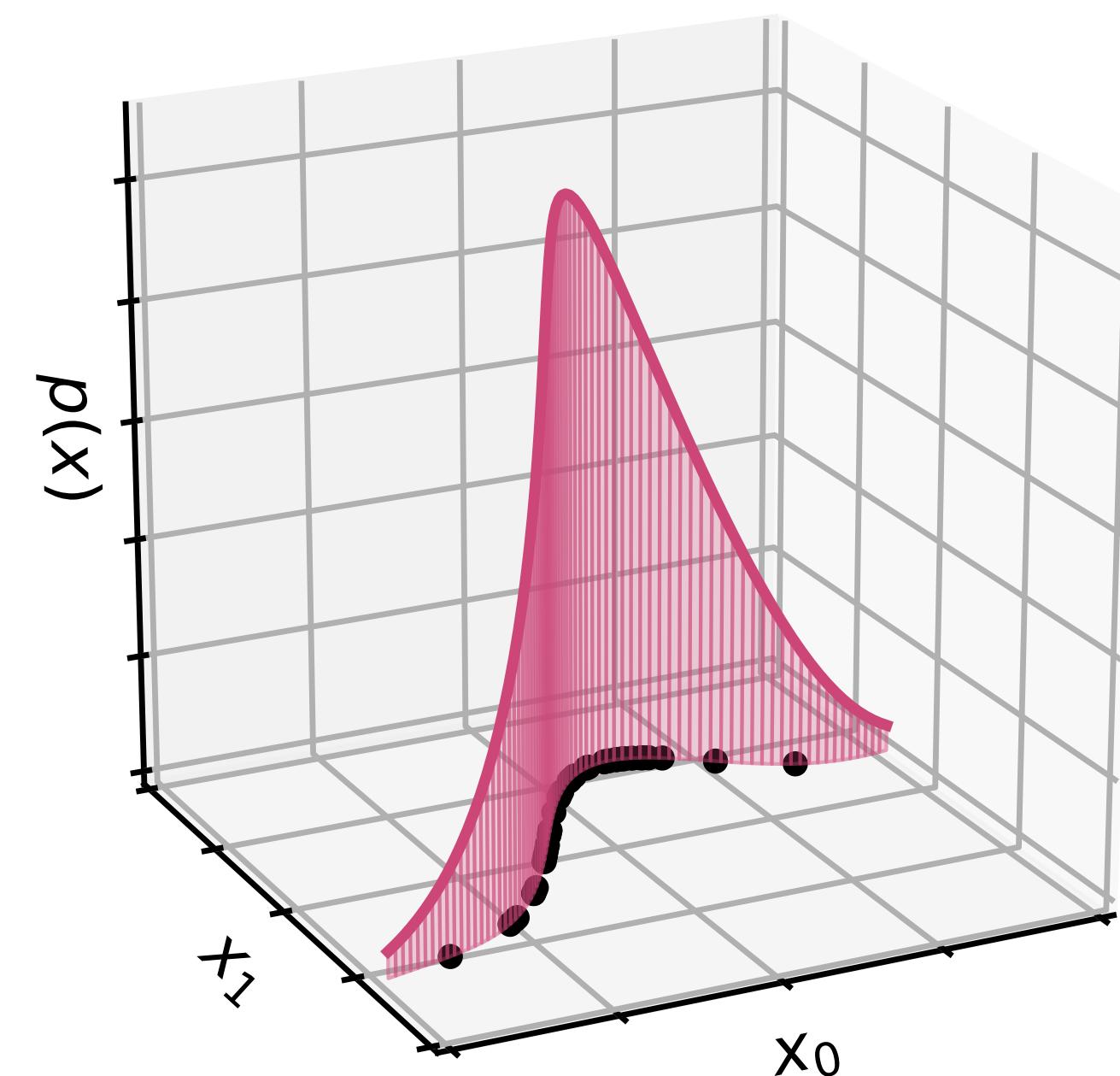
\mathcal{M} -flows

[JB, K. Cranmer 2003.13913]

Often data is restricted to a lower-dimensional manifold embedded in the data space

\mathcal{M} -flows are a new probabilistic / generative model that

- describe data as a tractable probability density on a lower-dimensional manifold
- learn manifold and density from data



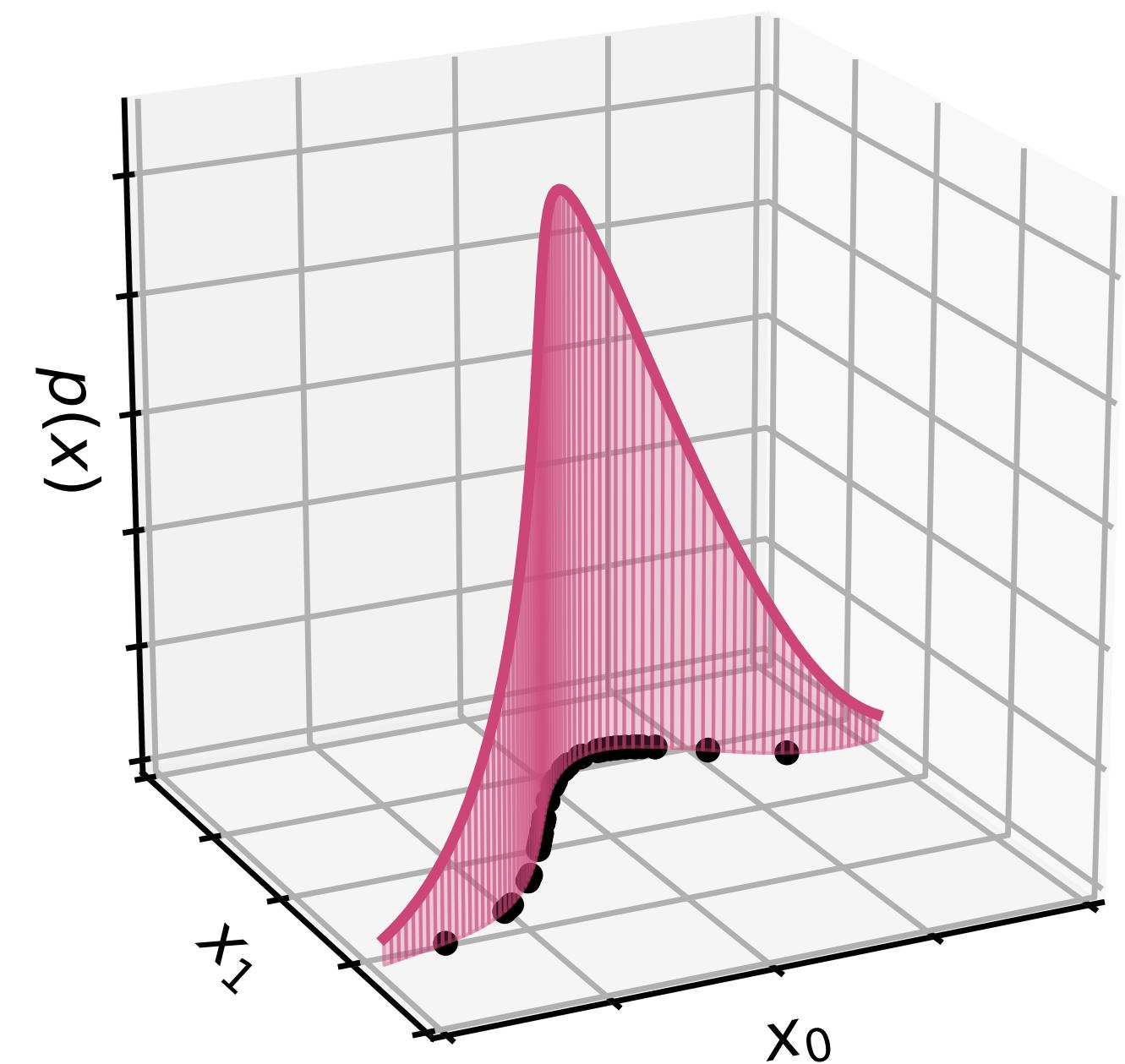
\mathcal{M} -flows

[JB, K. Cranmer 2003.13913]

Often data is restricted to a lower-dimensional manifold embedded in the data space

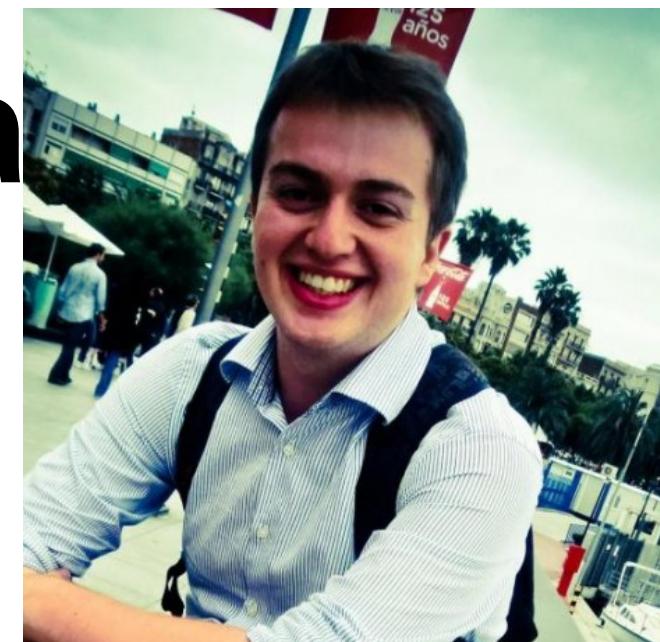
\mathcal{M} -flows are a new probabilistic / generative model that

- describe data as a tractable probability density on a lower-dimensional manifold
- learn manifold and density from data





Kyle Cranmer



Gilles Louppe



Juan Pavez



Markus Stoye



Felix Kling



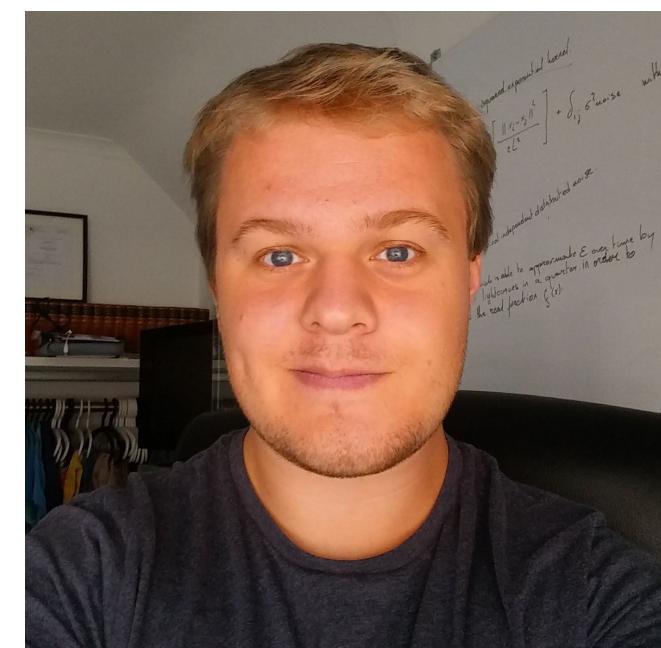
Irina Espejo



Sinclert Perez



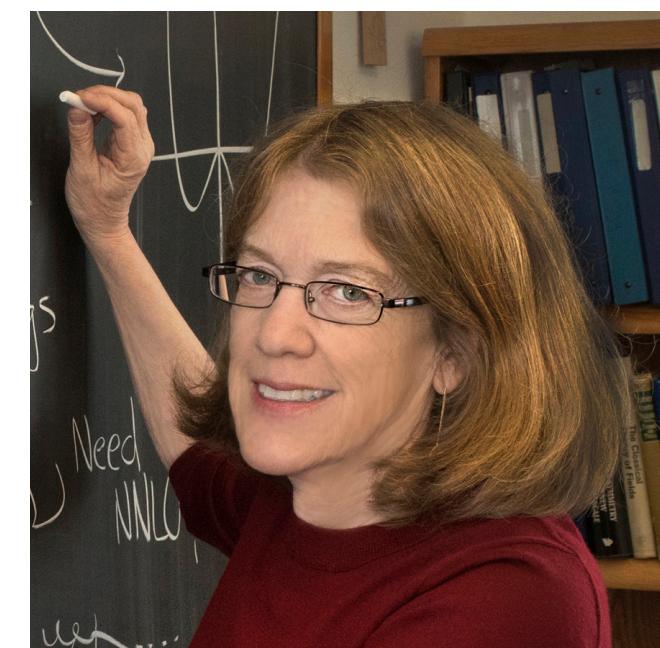
Sid Mishra-Sharma



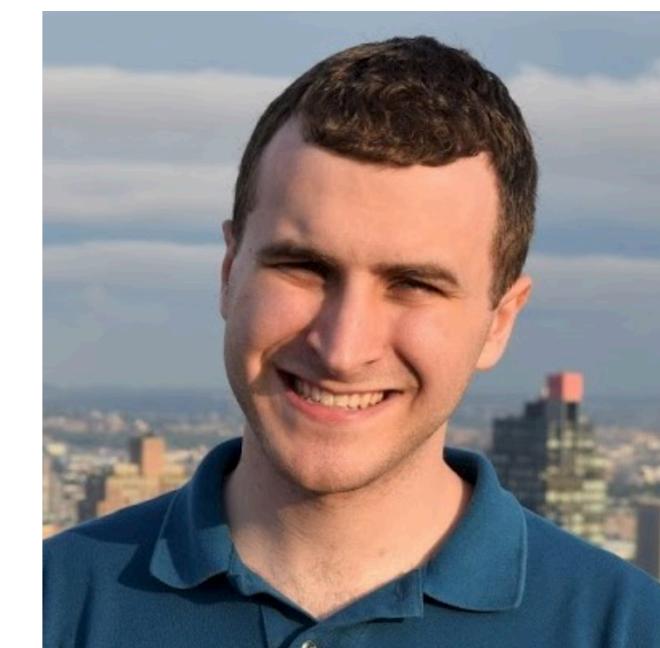
Joeri Hermans



Tilman Plehn



Sally Dawson



Sam Homiller

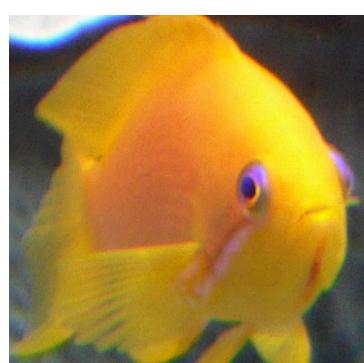


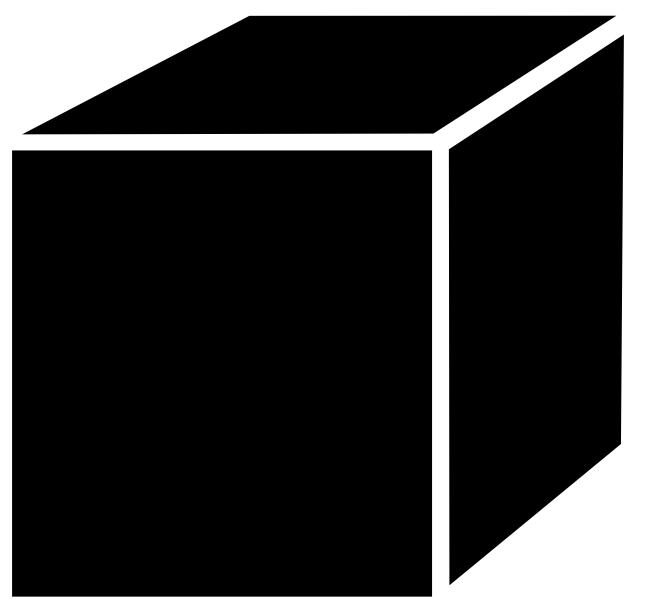
Zubair Bhatti

Parts of this talk were inspired by great presentations by Kyle Cranmer, Gilles Louppe, Sid Mishra-Sharma, and Jakob Macke

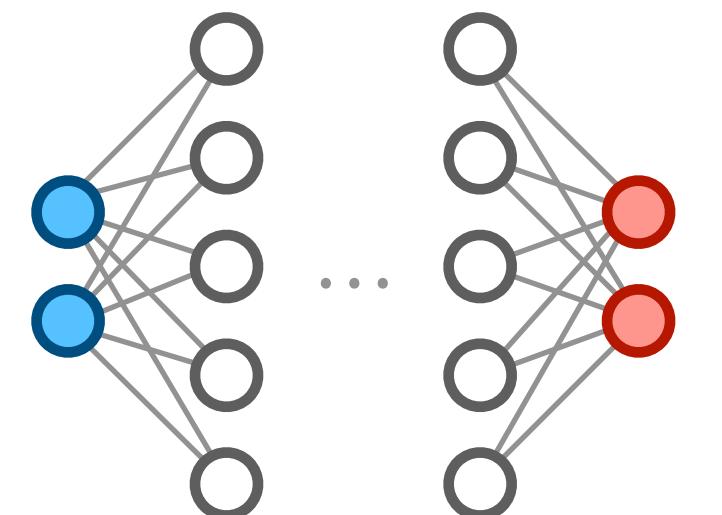


The SCAILFIN Project
scailfin.github.io

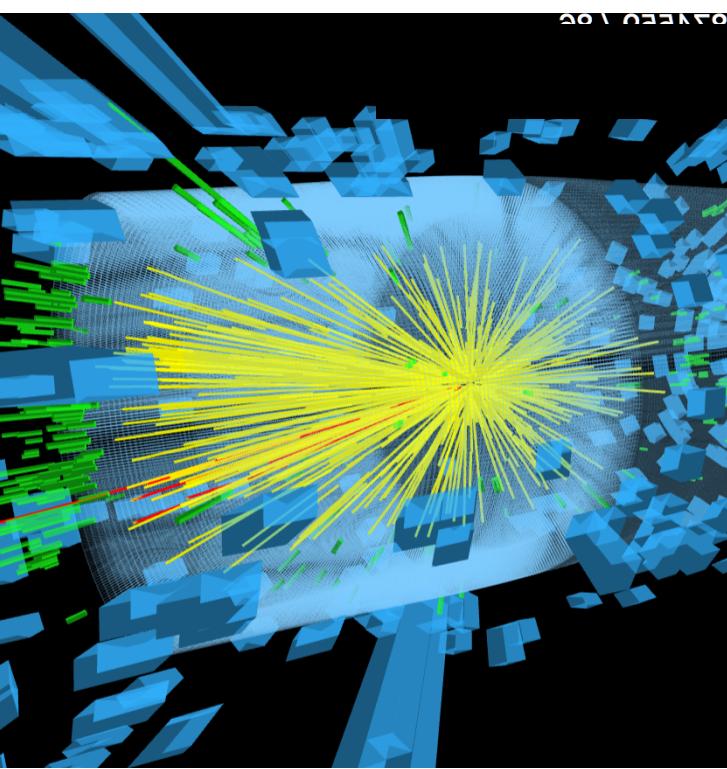




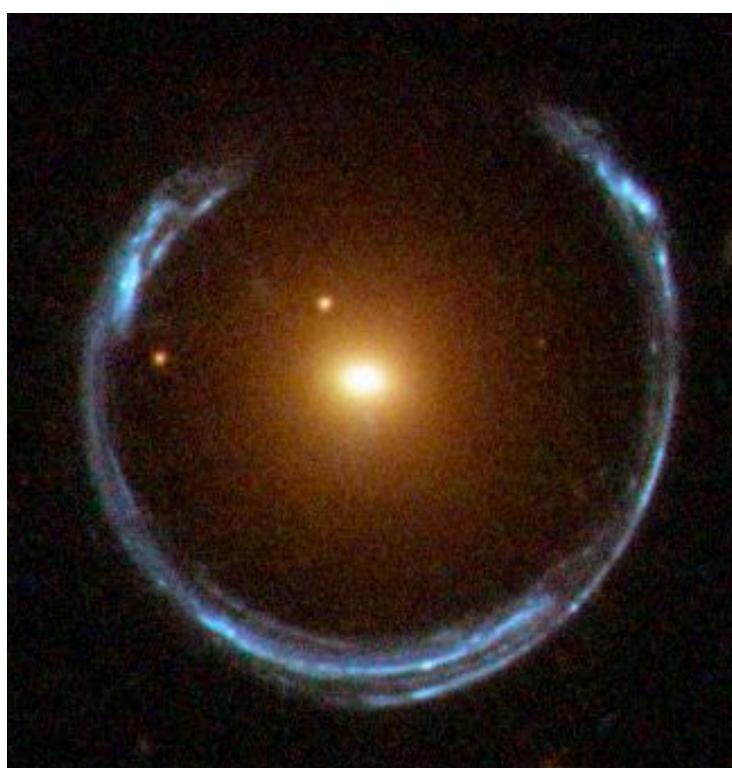
Simulators make precise predictions, but inference is challenging.



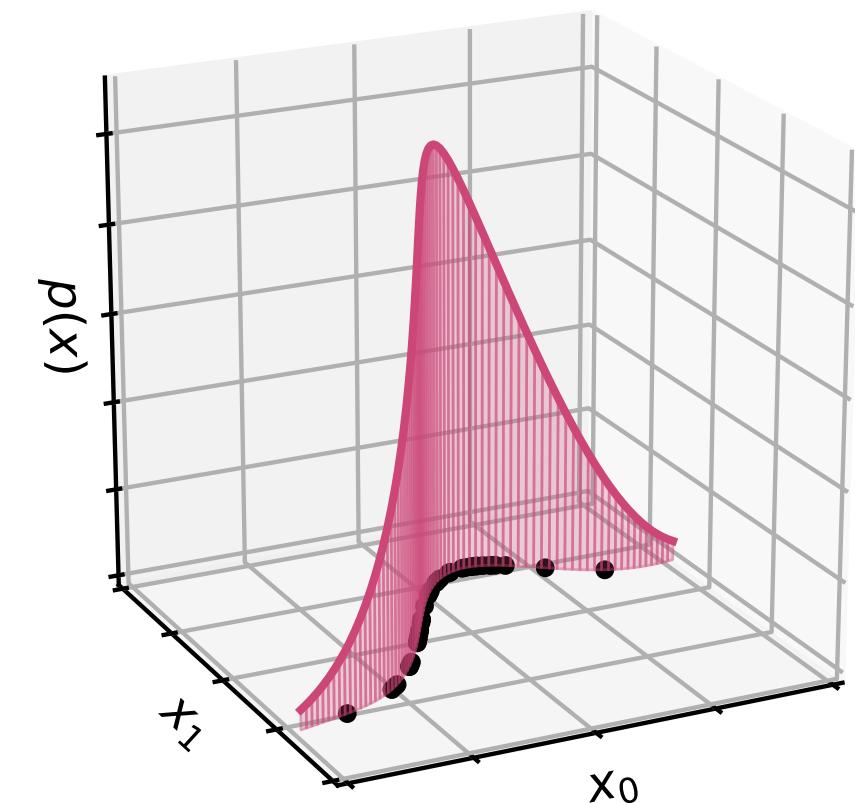
Machine learning enables powerful inference methods, especially when we inject more information.



They work in problems from the smallest...



... to the largest scales.



Further ML advances will translate into scientific progress.

Selected references

Review of simulation-based inference

K. Cranmer, **J. Brehmer**, G. Louppe:
“The frontier of simulation-based inference”
PNAS, 1911.01429

Machine learning–based inference methods

J. Brehmer, G. Louppe, J. Pavez, K. Cranmer:
“Mining gold from implicit models to improve likelihood-free inference”
PNAS, 1805.12244

M. Stoye, **J. Brehmer**, K. Cranmer, G. Louppe, J. Pavez:
“Likelihood-free inference with an improved cross-entropy estimator”
NeurIPS workshop, 1808.00973

Probabilistic models

J. Brehmer and K. Cranmer:
“Flows for simultaneous manifold learning and density estimation”
ICML workshop, 2003.13913

Particle physics

J. Brehmer, K. Cranmer, G. Louppe, J. Pavez:
“Constraining Effective Field Theories with machine learning”
PRL, 1805.00013

J. Brehmer, K. Cranmer, G. Louppe, J. Pavez:
“A guide to constraining Effective Field Theories with machine learning”
PRD, 1805.00020

J. Brehmer, F. Kling, I. Espejo, K. Cranmer:
“MadMiner: Machine learning–based inference for particle physics”
CSBS, 1907.10621, <https://github.com/diana-hep/madminer>

J. Brehmer, K. Cranmer, F. Kling, and T. Plehn:
“Better Higgs Measurements Through Information Geometry”
PRD, 1612.05261

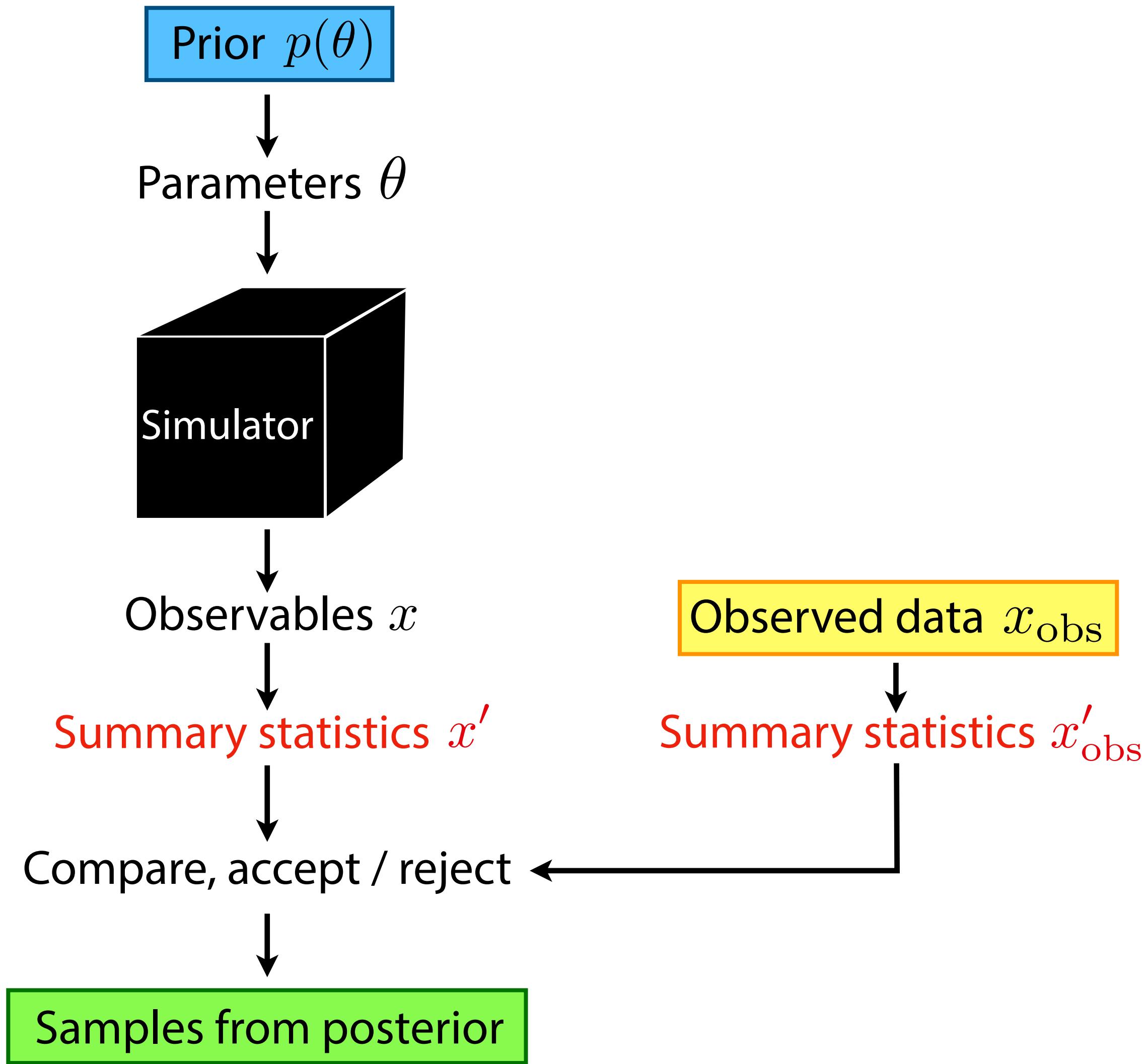
Astrophysics

J. Brehmer, S. Mishra-Sharma, J. Hermans, G. Louppe, K. Cranmer
“Mining for Dark Matter Substructure: Inferring subhalo population properties from strong lenses with machine learning”
ApJ, 1909.02005

Bonus material: simulation-based inference

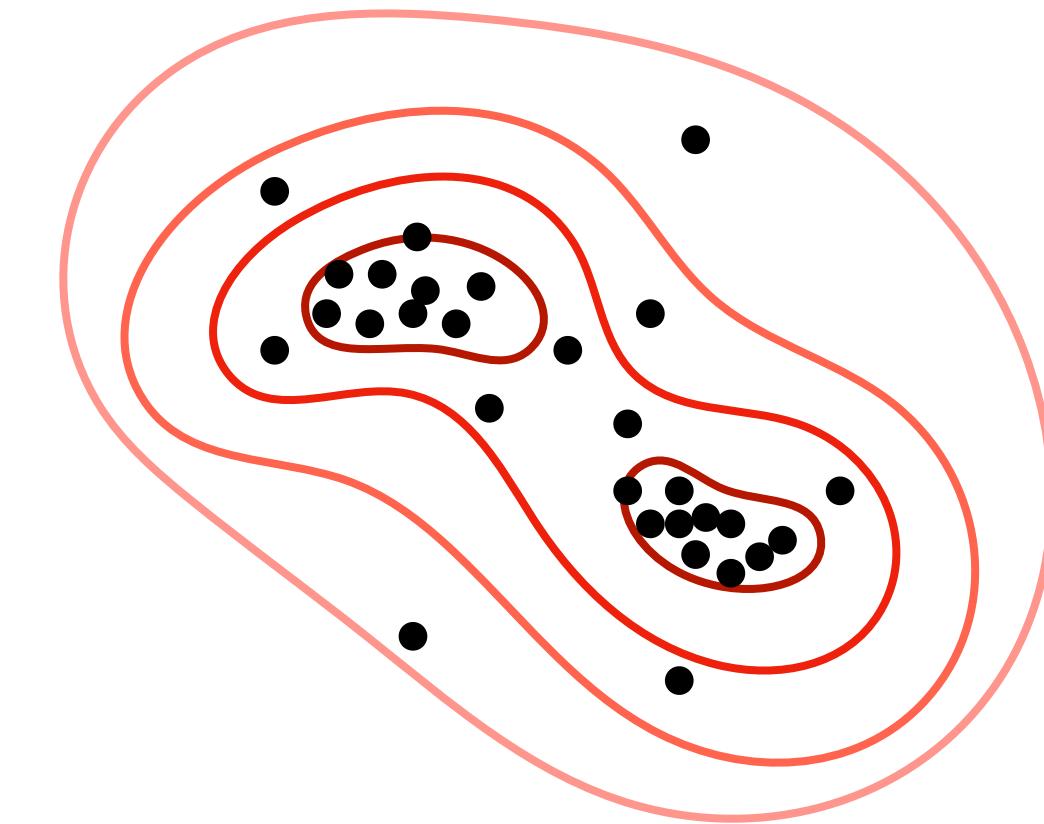
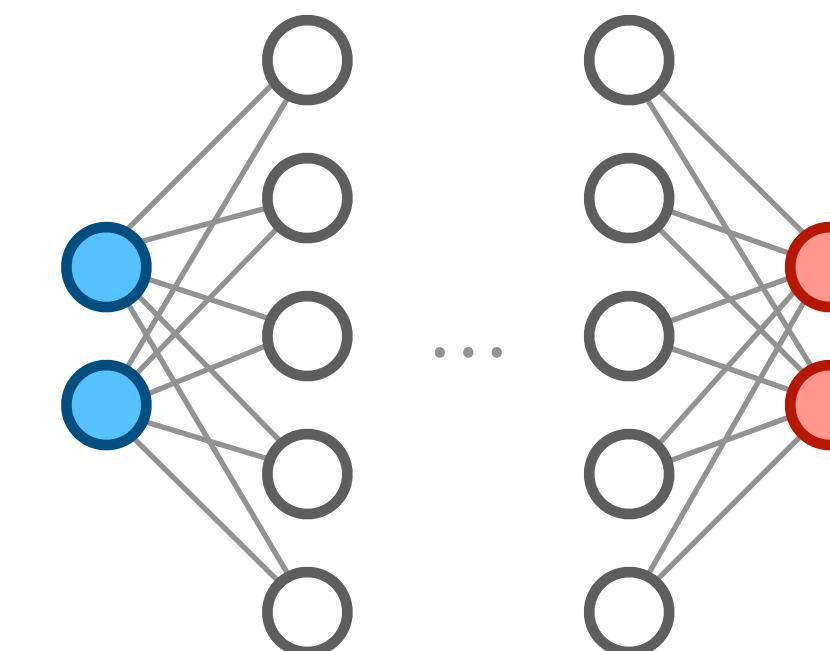
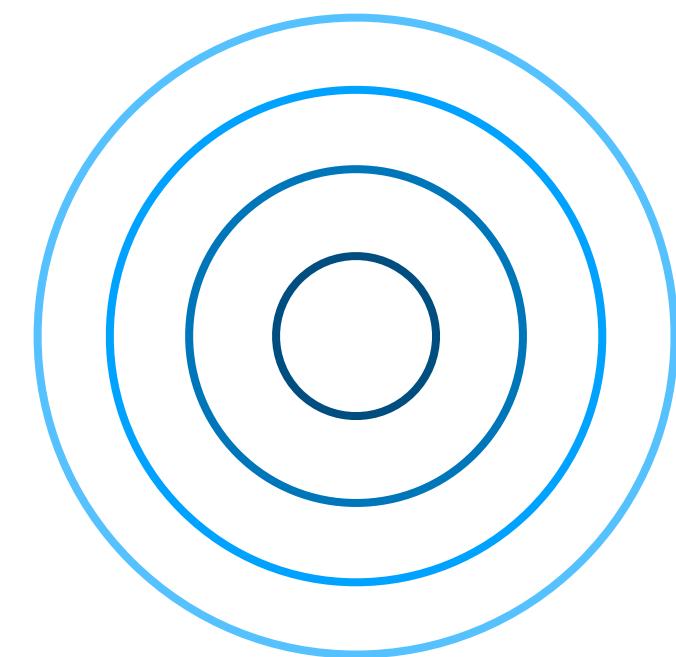
Approximate Bayesian Computation (ABC)

[D. Rubin 1984]



- Compression to summary statistics and acceptance threshold reduce quality of inference
- Rejection algorithm can be very sample inefficient

High-dimensional density estimation with normalizing flows



Simple base density

$$u \sim \pi(u)$$

NN: transformation $x = f(u)$

- one-to-one and invertible
- differentiable
- f^{-1} and $\det \nabla f$ are tractable

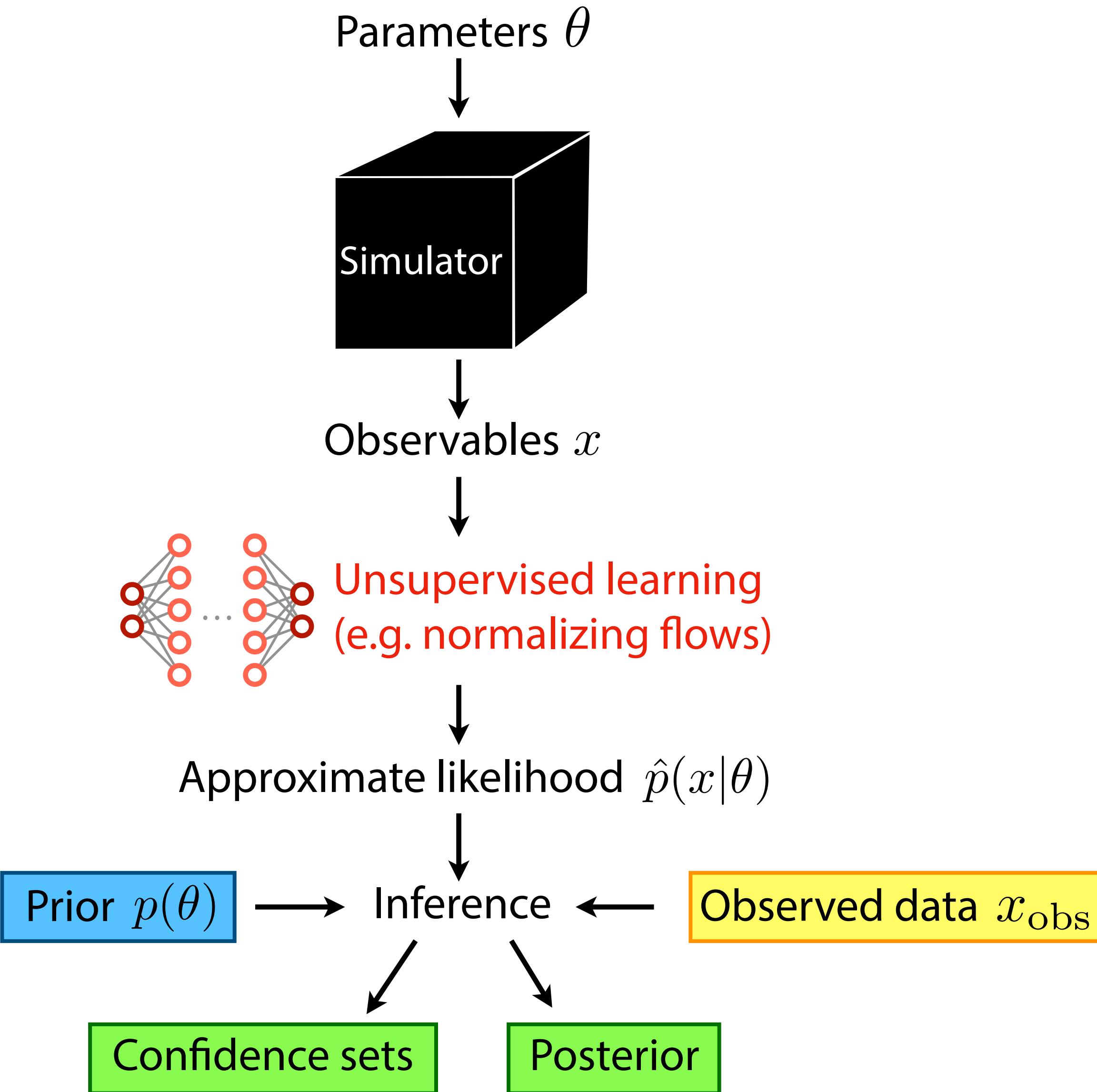
Target density is given by

$$\hat{p}(x) = \pi(f^{-1}(x)) |\det \nabla f|^{-1}$$

Train transformation by
maximizing $\log \hat{p}(x)$

Transformation can depend on θ
to model conditional density $\log \hat{p}(x|\theta)$

Inference with neural likelihood estimation



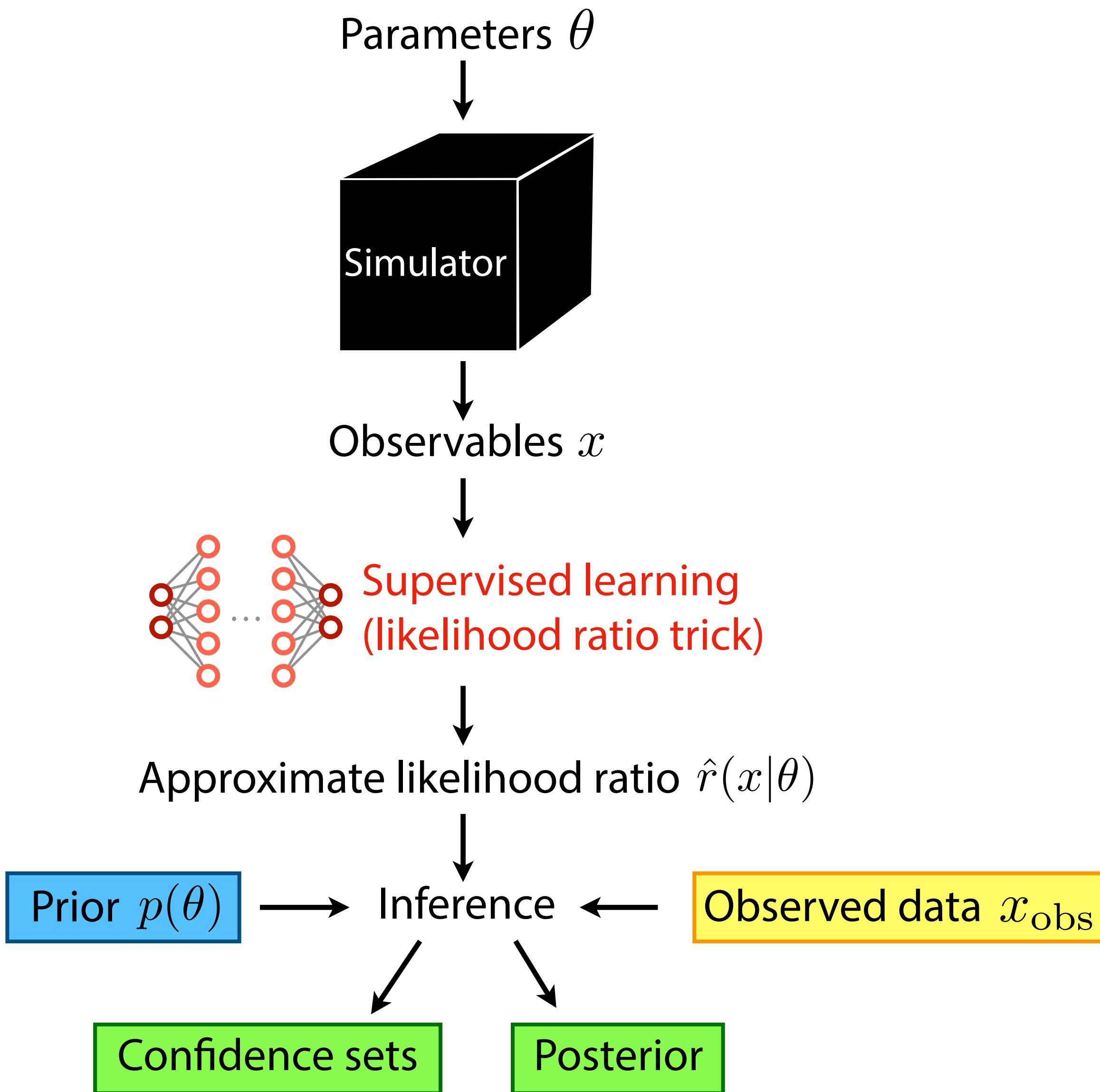
[G. Papamakarios, D. Sterratt, I. Murray 1805.07226;
J.-M. Lueckmann, G. Bassetto, T. Karaletsos, J. Macke 1805.09294]

- Conditional neural density estimator (e.g. normalizing flow) as tractable surrogate for simulator likelihood
- Scales well to high-dimensional data (no compression to summary stats necessary)
- Amortized: After upfront simulation + training phase, inference is efficient for new data or prior
- Related alternative: learn posterior $\hat{p}(\theta|x_{\text{obs}})$

[G. Papamakarios et al 1605.06376;
J.-M. Lueckmann et al 1711.01861]

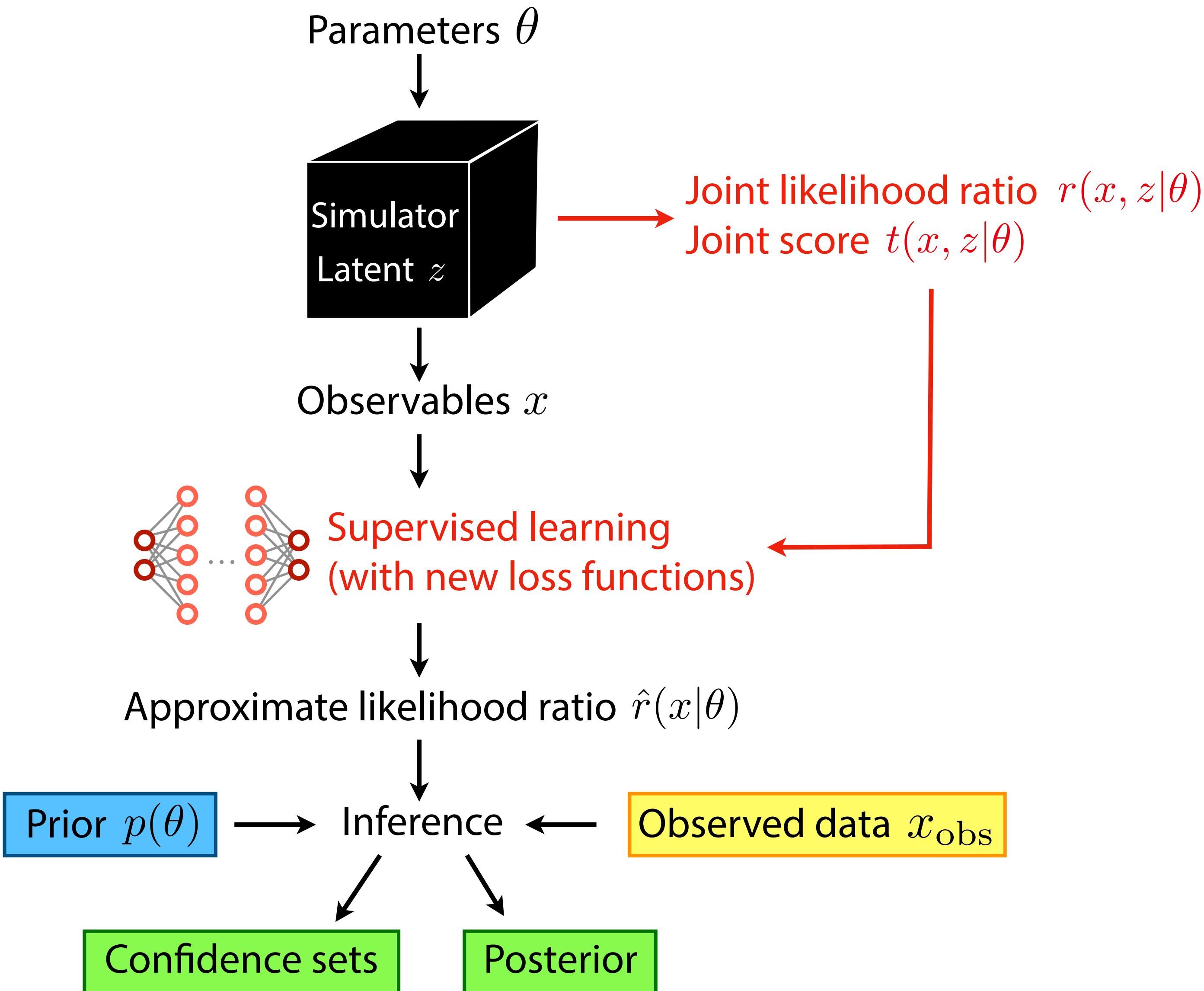
Inference by likelihood ratio trick

[K. Cranmer J. Pavez, G. Louppe 1506.02169]

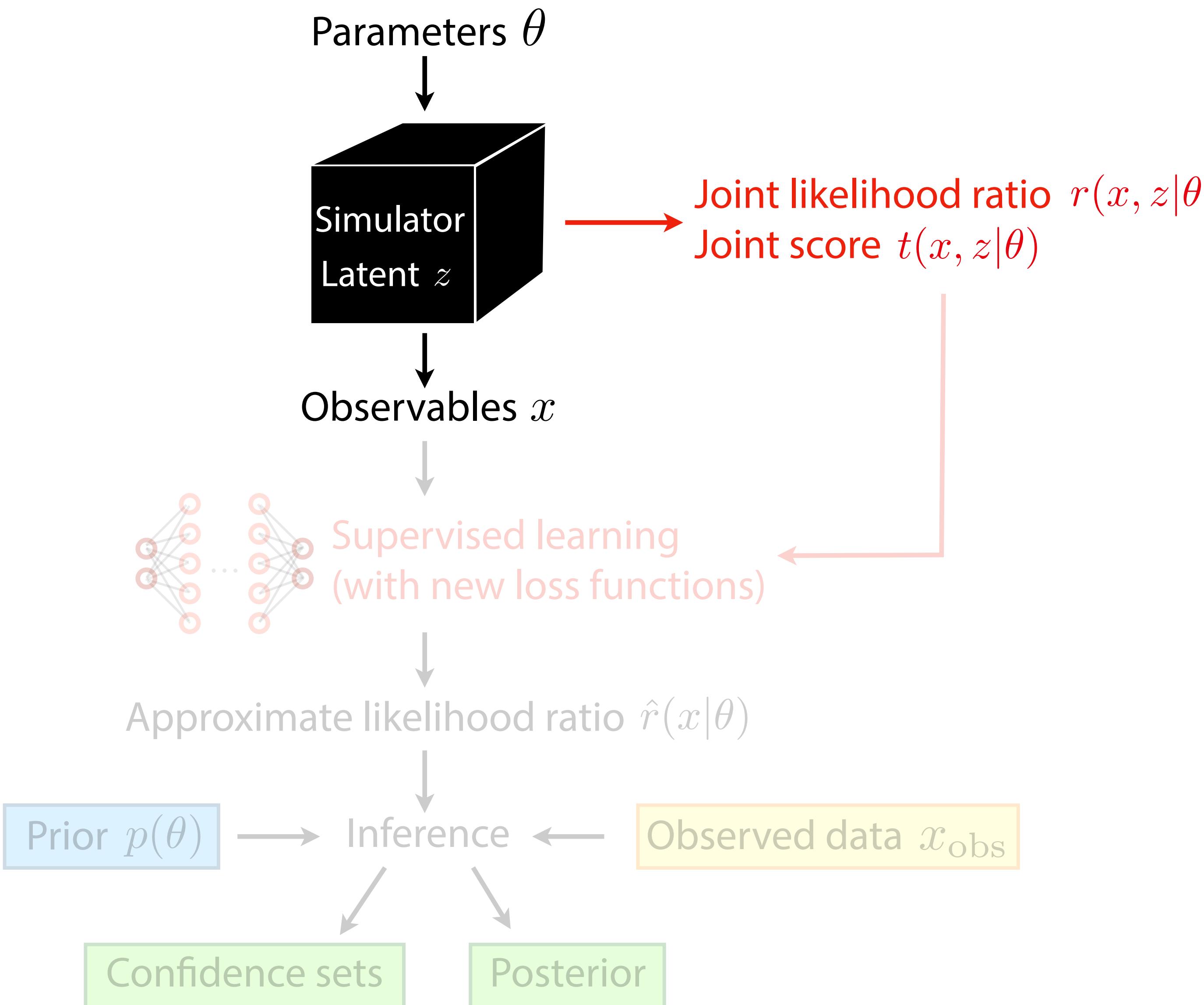


- For inference, likelihood and likelihood ratio are interchangeable
- Advantage: Learning the likelihood ratio can be a simpler task than learning the likelihood
- Disadvantage: Cannot sample from likelihood ratio

Mining gold



Step 1: Extracting more information from simulations

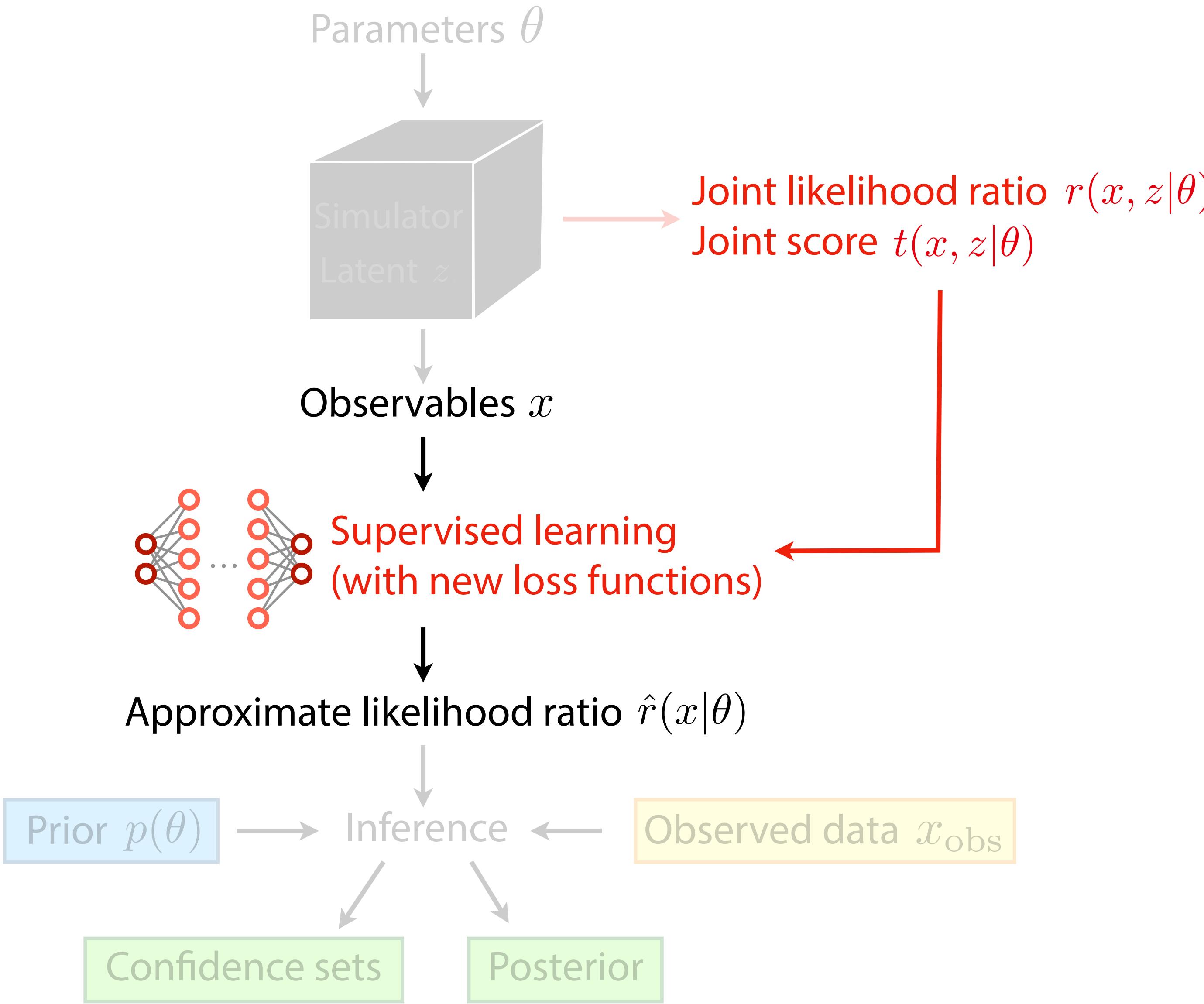


For each simulated event, calculate

- joint likelihood ratio $r(x, z|\theta) = \frac{p(x, z|\theta)}{p_{\text{ref}}(x, z)}$
- joint score $t(x, z|\theta) = \nabla_{\theta} \log p(x, z|\theta)$

How much more or less likely would this simulated sample (fixing all latent variables) be when changing the parameters?

Step 2: Machine learning



- Train a neural network $g(x, \theta)$ on loss functionals like

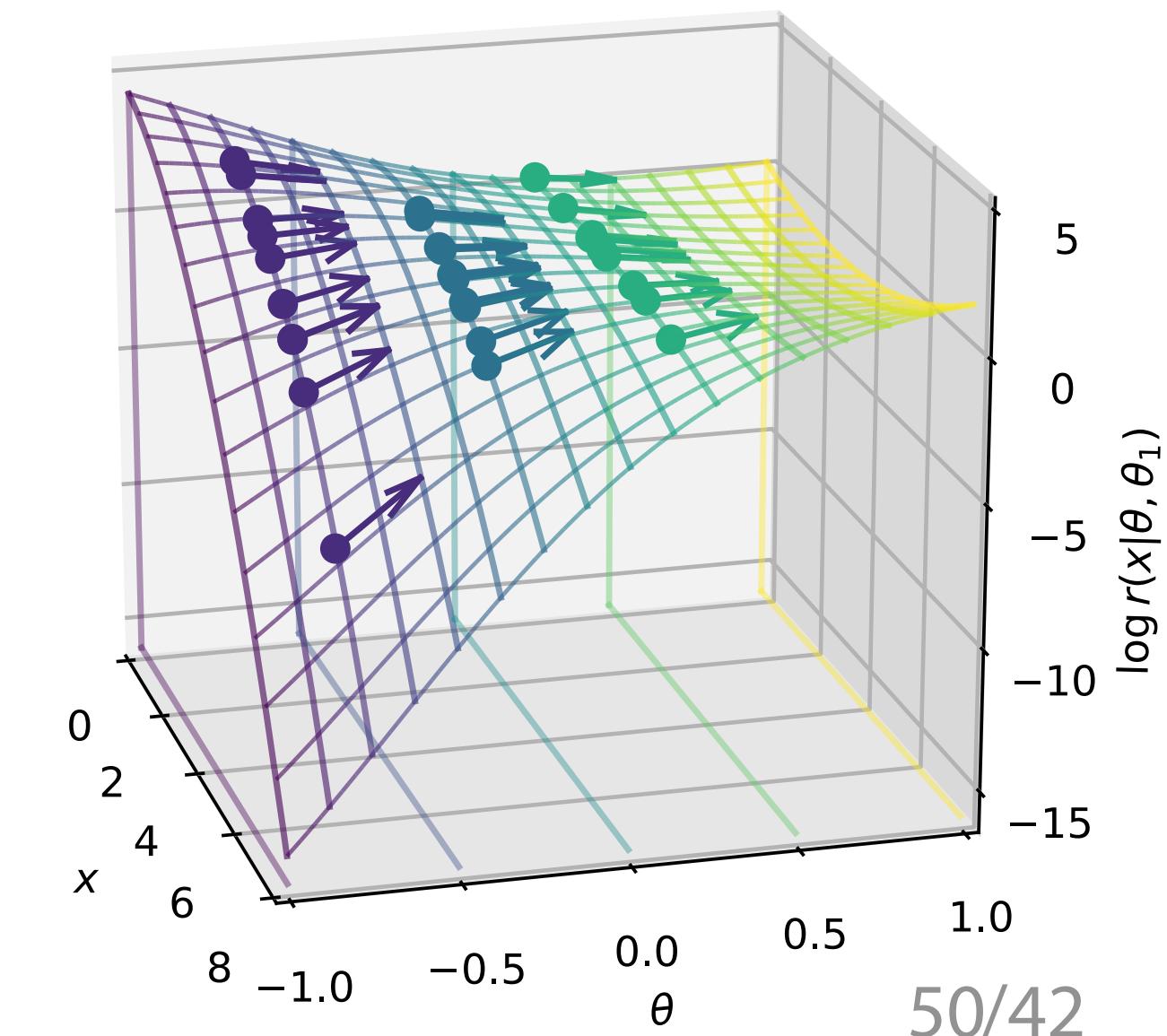
$$L[g] = \frac{1}{N} \sum_i |g(x_i, \theta_i) - r(x_i, z_i|\theta_i)|^2$$

- The network will converge to

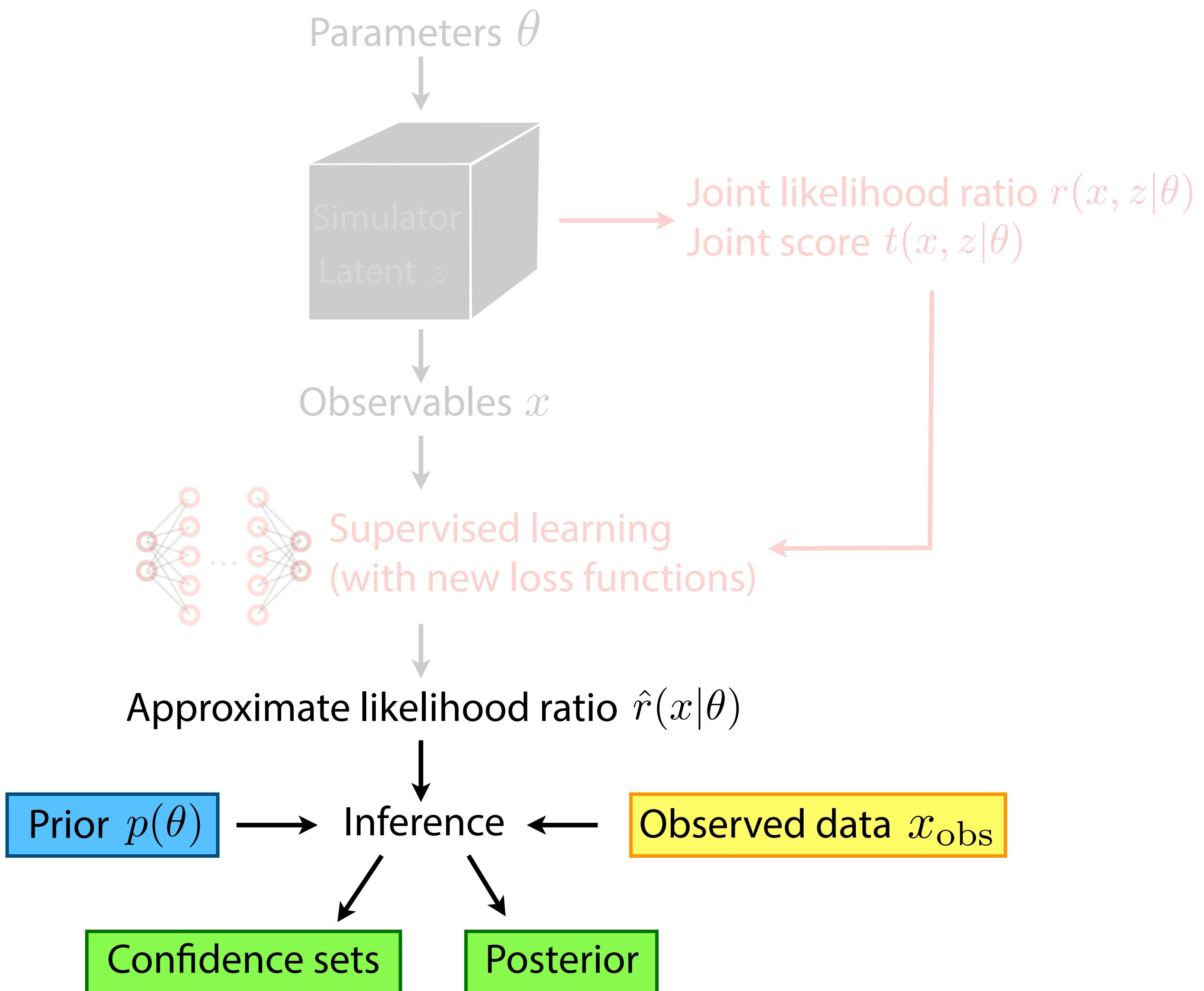
$$g(x, \theta) \rightarrow \arg \min L[g] = r(x|\theta) = \frac{p(x|\theta)}{p_{\text{ref}}(x)} !$$

(for sufficient training data, NN capacity, efficient optimization)

- RASCAL:
Joint score adds gradient information
 \Rightarrow three orthogonal pieces of information



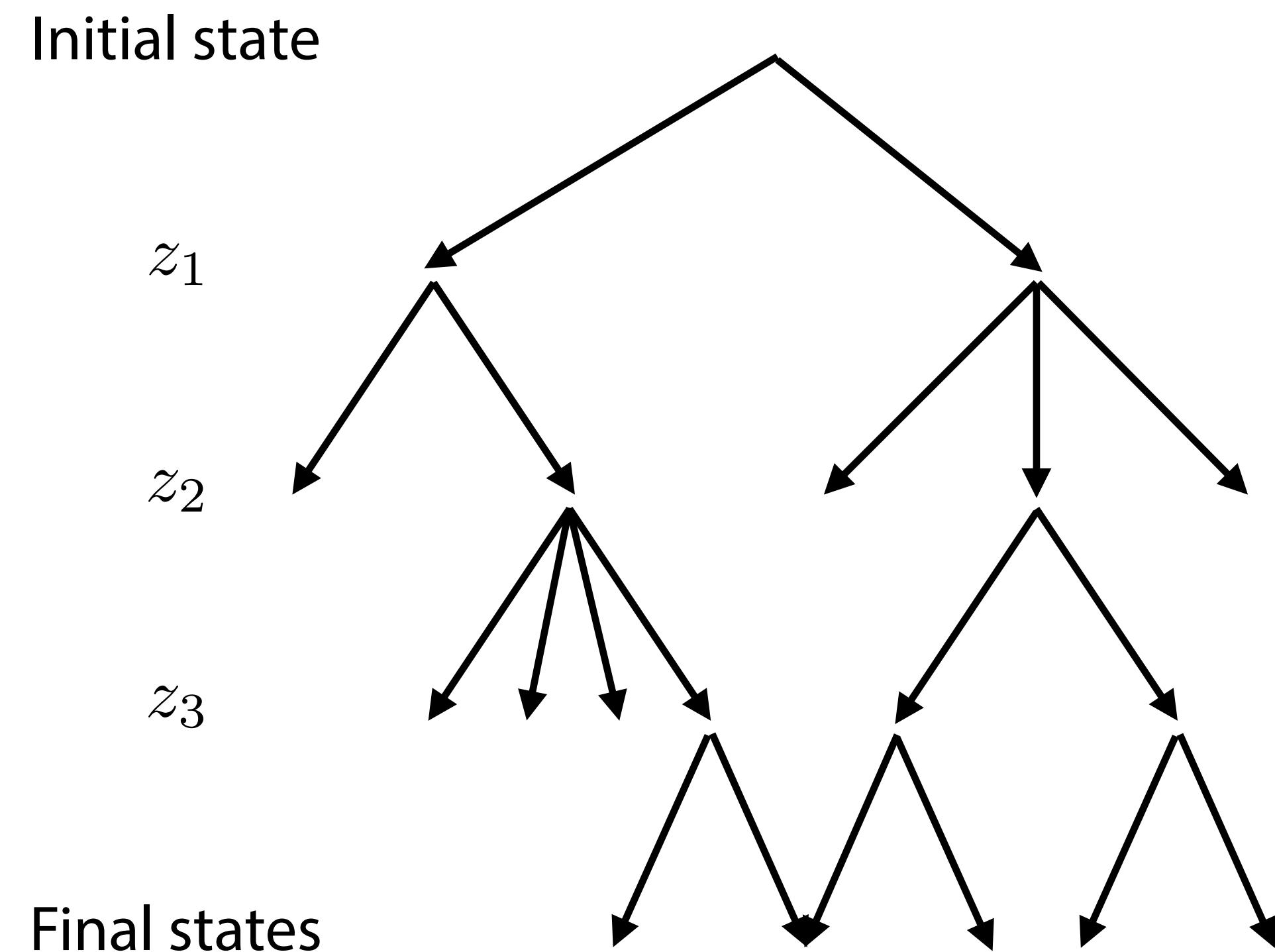
Step 3: Inference



Mining gold from any simulation

- Computer simulation typically evolve along a tree-like structure of successive random branchings
- The probabilities of each branching $p_i(z_i|z_{i-1}, \theta)$ are often clearly defined in the code:

```
if random() > 0.1 + 2.5 * model_parameter:  
    do_one_thing()  
else:  
    do_another_thing()
```



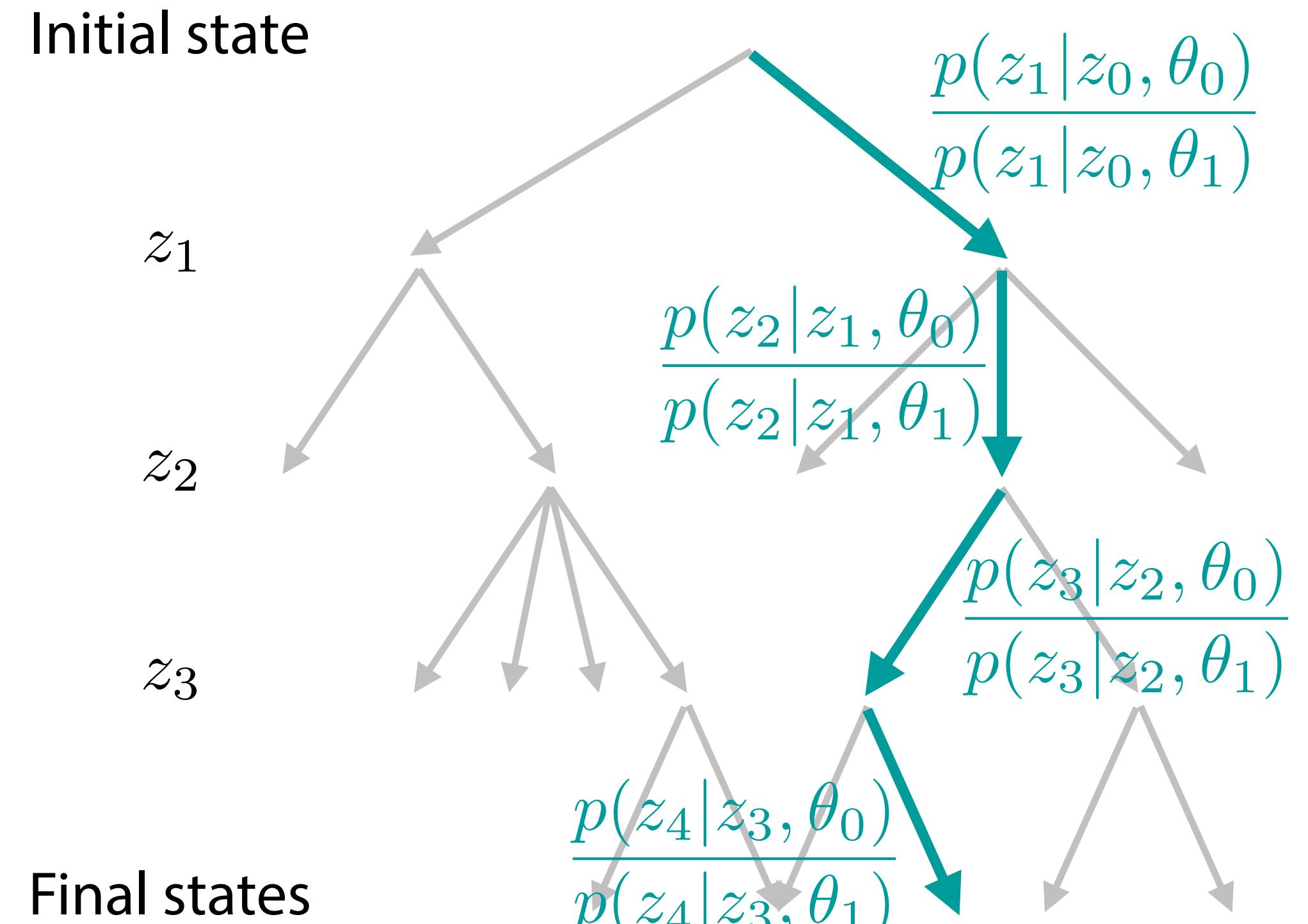
Mining gold from any simulation

- Computer simulation typically evolve along a tree-like structure of successive random branchings
- The probabilities of each branching $p_i(z_i|z_{i-1}, \theta)$ are often clearly defined in the code:

```
if random() > 0.1 + 2.5 * model_parameter:  
    do_one_thing()  
else:  
    do_another_thing()
```

- For each run of the simulator, we can calculate the probability **of the chosen path** for different values of the parameters, and the “**joint likelihood ratio**”:

$$r(x, z|\theta_0, \theta_1) = \frac{p(x, z|\theta_0)}{p(x, z|\theta_1)} = \prod_i \frac{p(z_i|z_{i-1}, \theta_0)}{p(z_i|z_{i-1}, \theta_1)}$$



The value of gold

Expectation value of the joint likelihood ratio:

$$\begin{aligned}\mathbb{E}_{z \sim p(z|x, \theta_1)} [\textcolor{teal}{r}(x, z|\theta_0, \theta_1)] &= \int dz p(z|x, \theta_1) \frac{p(x, z|\theta_0)}{p(x, z|\theta_1)} \\ &= \int dz \frac{p(x, z|\theta_1)}{p(x|\theta_1)} \frac{p(x, z|\theta_0)}{p(x, z|\theta_1)} \\ &= \textcolor{red}{r}(x|\theta_0, \theta_1)\end{aligned}$$

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

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

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.)

Machine learning = applied calculus of variations

So to get a good estimator of the likelihood ratio, we need to minimize a functional numerically:

Variational family Extremization Functional with integral

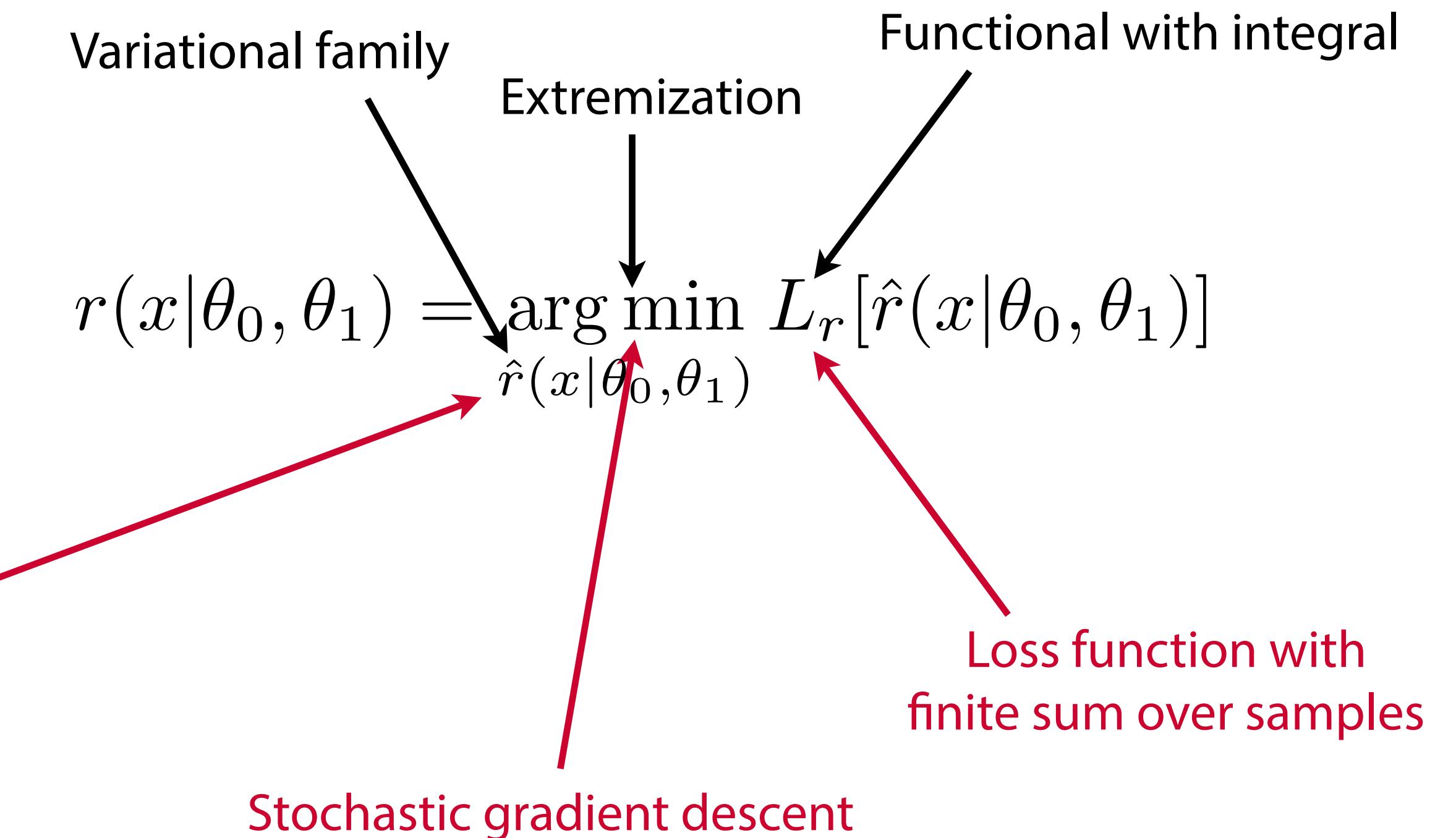
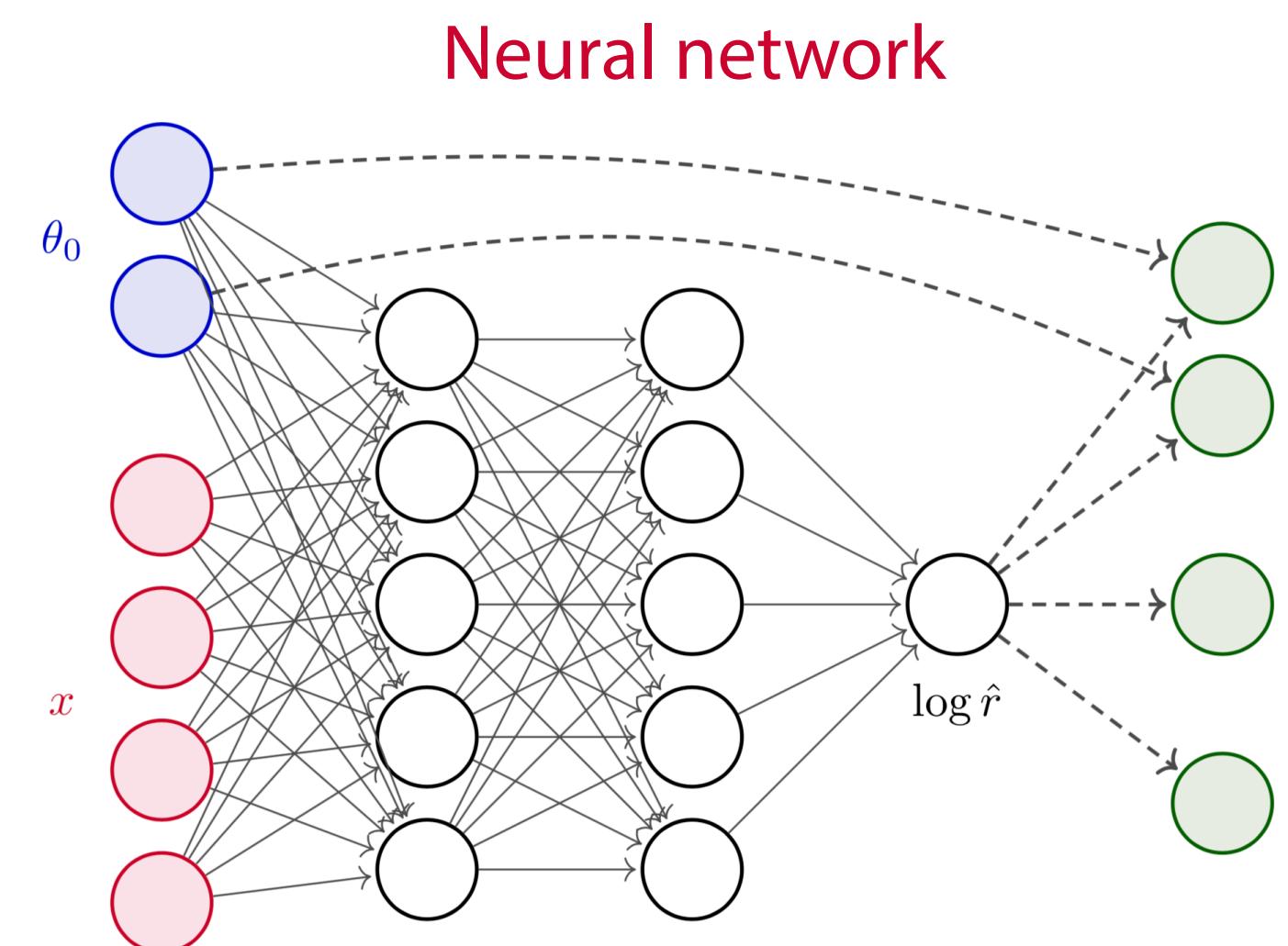
$$r(x|\theta_0, \theta_1) = \arg \min_{\hat{r}(x|\theta_0, \theta_1)} L_r[\hat{r}(x|\theta_0, \theta_1)]$$

A sufficiently expressive neural network
efficiently trained in this way
with enough data will learn
the likelihood ratio function $r(x|\theta_0, \theta_1)$!

Machine learning = applied calculus of variations

So to get a good estimator of the likelihood ratio, we need to minimize a functional numerically:

This is where machine learning comes in!



A sufficiently expressive neural network efficiently trained in this way with enough data will learn the likelihood ratio function $r(x|\theta_0, \theta_1)$!

The local model

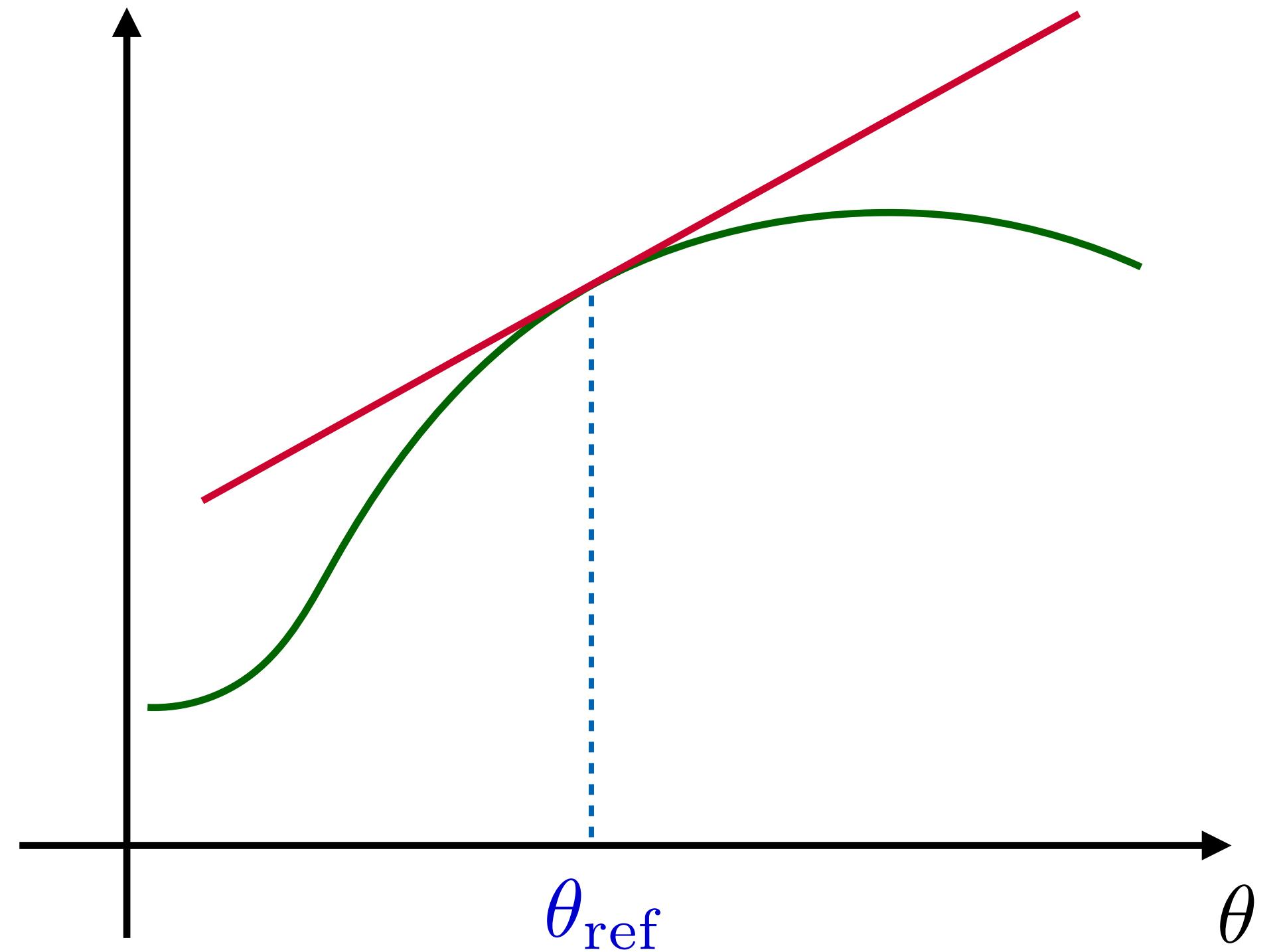
[see also J. Alsing, B. Wandelt 1712.00012; J. Alsing, B. Wandelt, S. Freeney 1801.01497;
P. de Castro, T. Dorigo 1806.04743; J. Alsing, B. Wandelt 1903.01473]

Taylor expansion of $\log p(x|\theta)$ around θ_{ref} :

$$\begin{aligned}\log p(x|\theta) &= \log p(x|\theta_{\text{ref}}) \\ &+ \underbrace{\nabla_{\theta} \log p(x|\theta) \Big|_{\theta_{\text{ref}}} \cdot (\theta - \theta_{\text{ref}})}_{\equiv t(x|\theta_{\text{ref}})} \\ &+ \mathcal{O}((\theta - \theta_{\text{ref}})^2)\end{aligned}$$

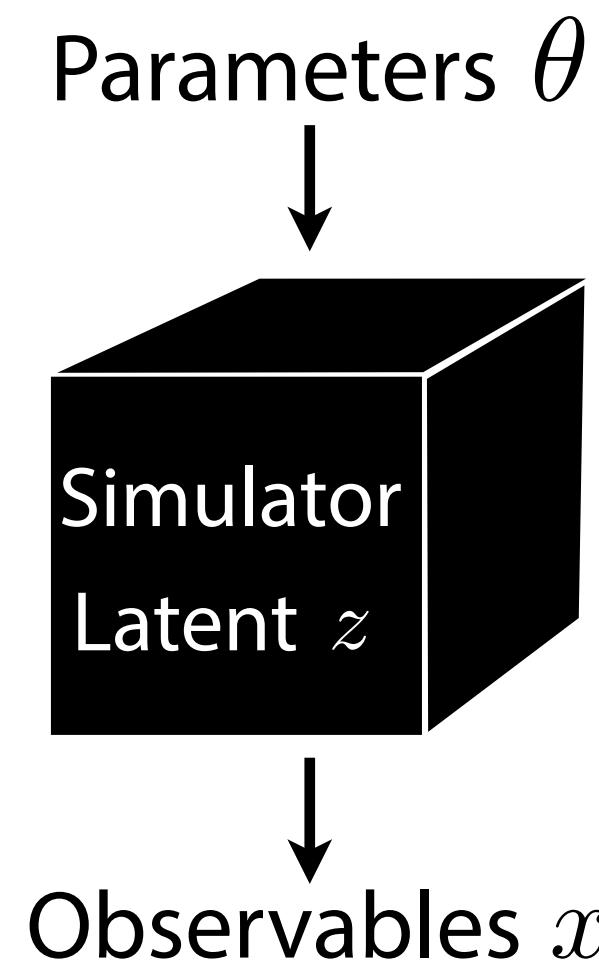
In the neighborhood of θ_{ref} :

- the **score vector** $t(x|\theta_{\text{ref}})$ is the sufficient statistics
- knowing $t(x|\theta_{\text{ref}})$ is just as powerful as knowing the full function $\log p(x|\theta)$
- $t(x|\theta_{\text{ref}})$ is the most powerful observable

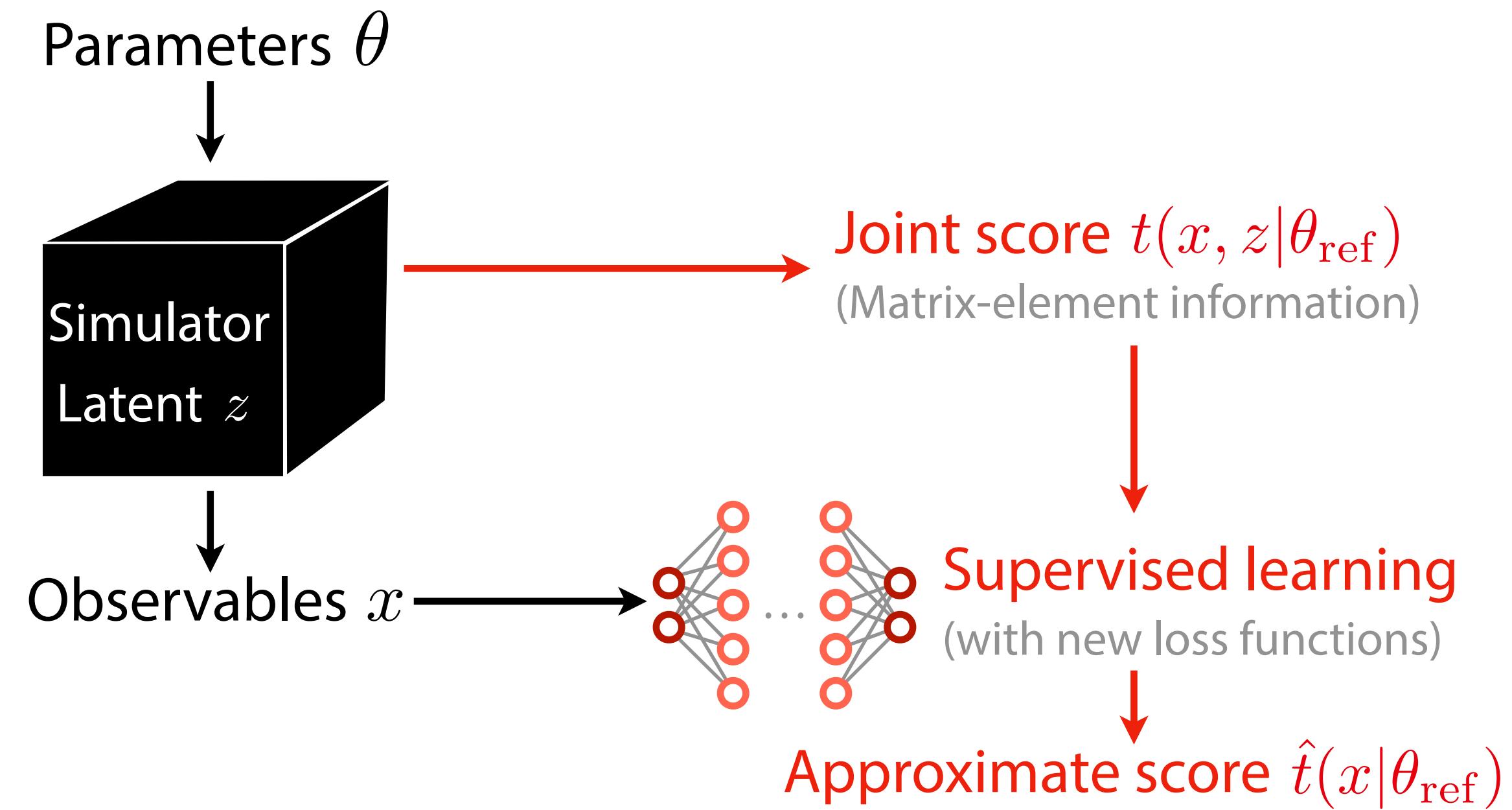


The score itself is intractable. But we can use the same trick as for the likelihood ratio!

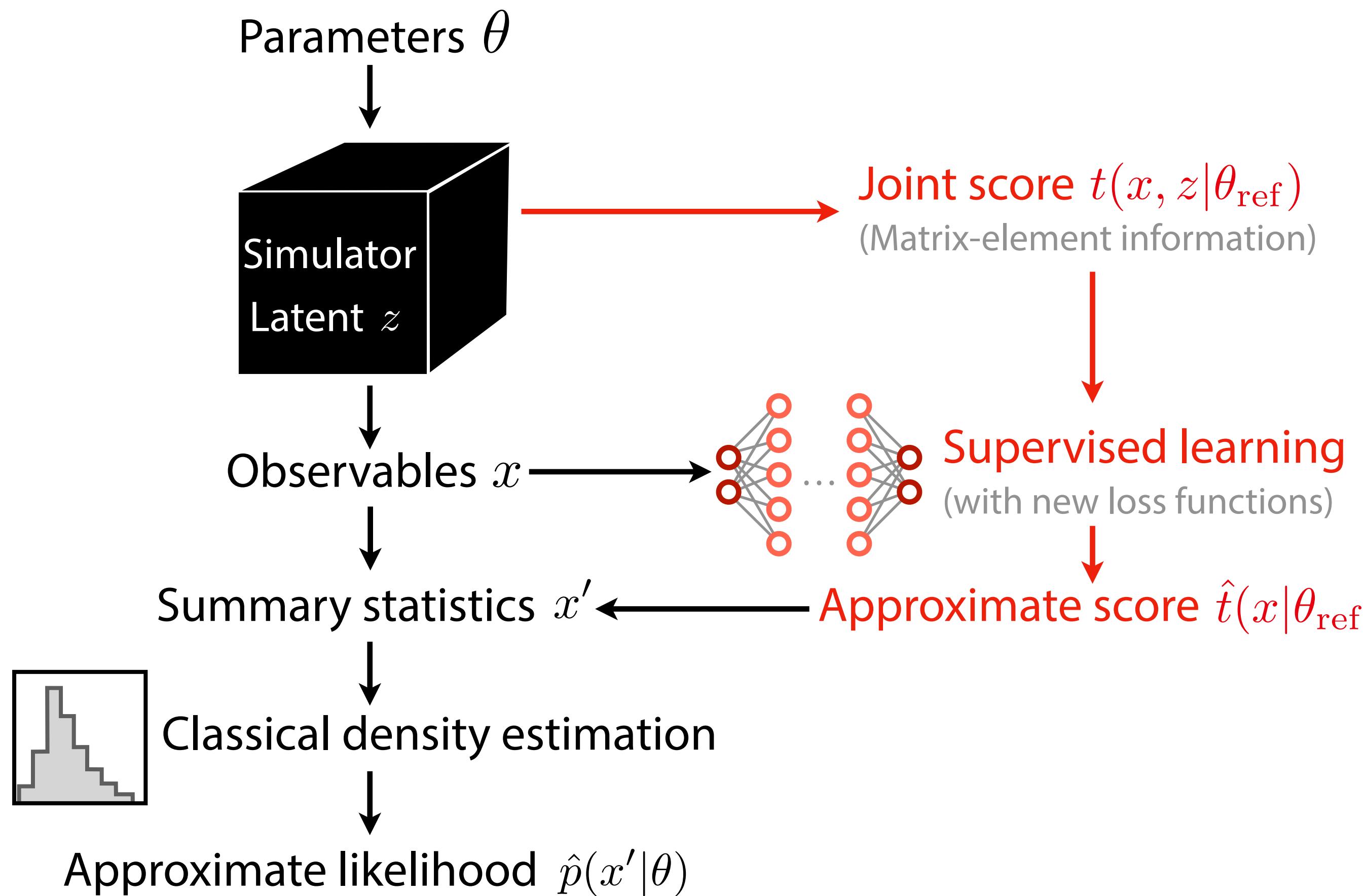
Neural optimal observables (SALLY)



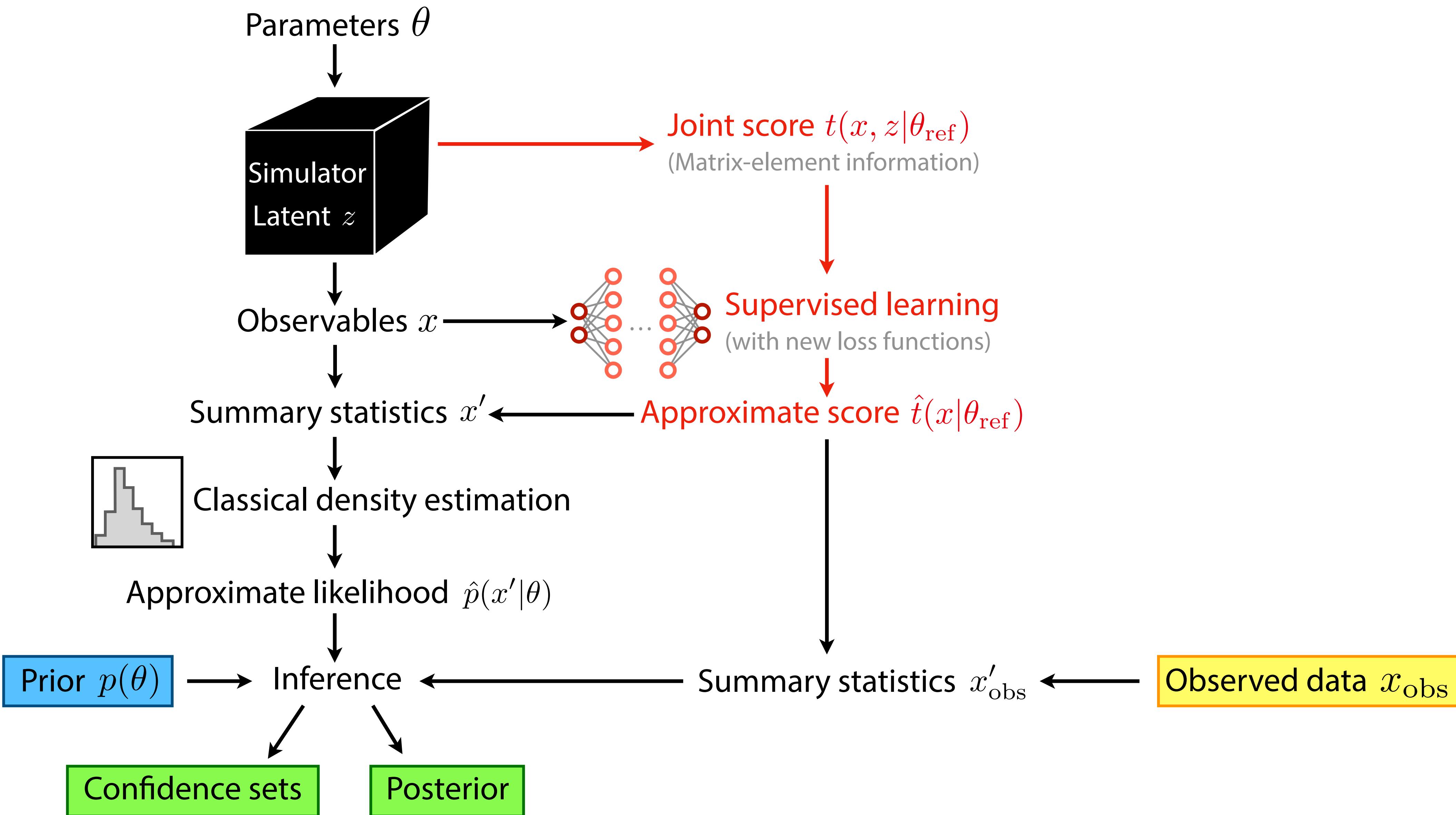
Neural optimal observables (SALLY)



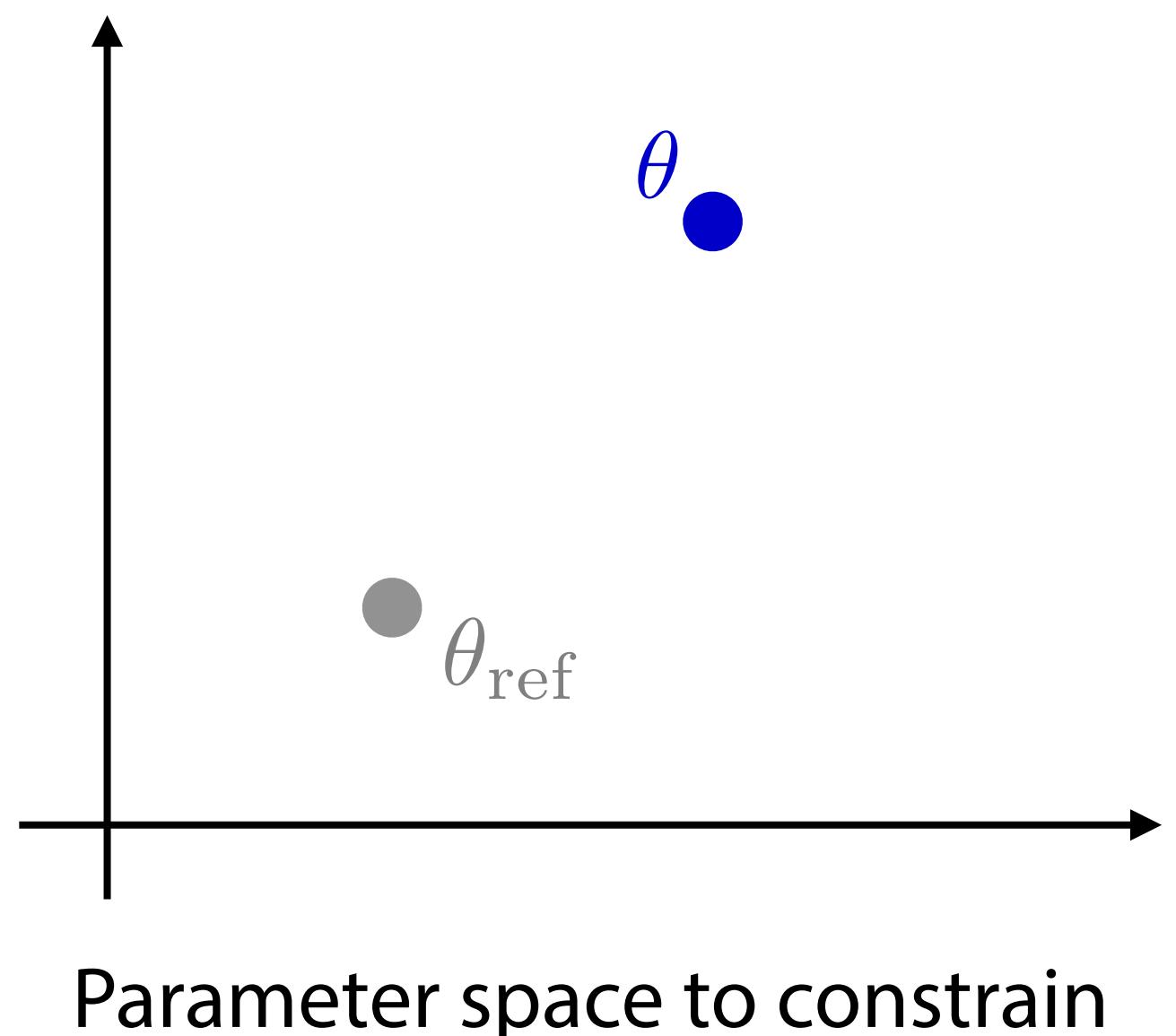
Neural optimal observables (SALLY)



Neural optimal observables (SALLY)

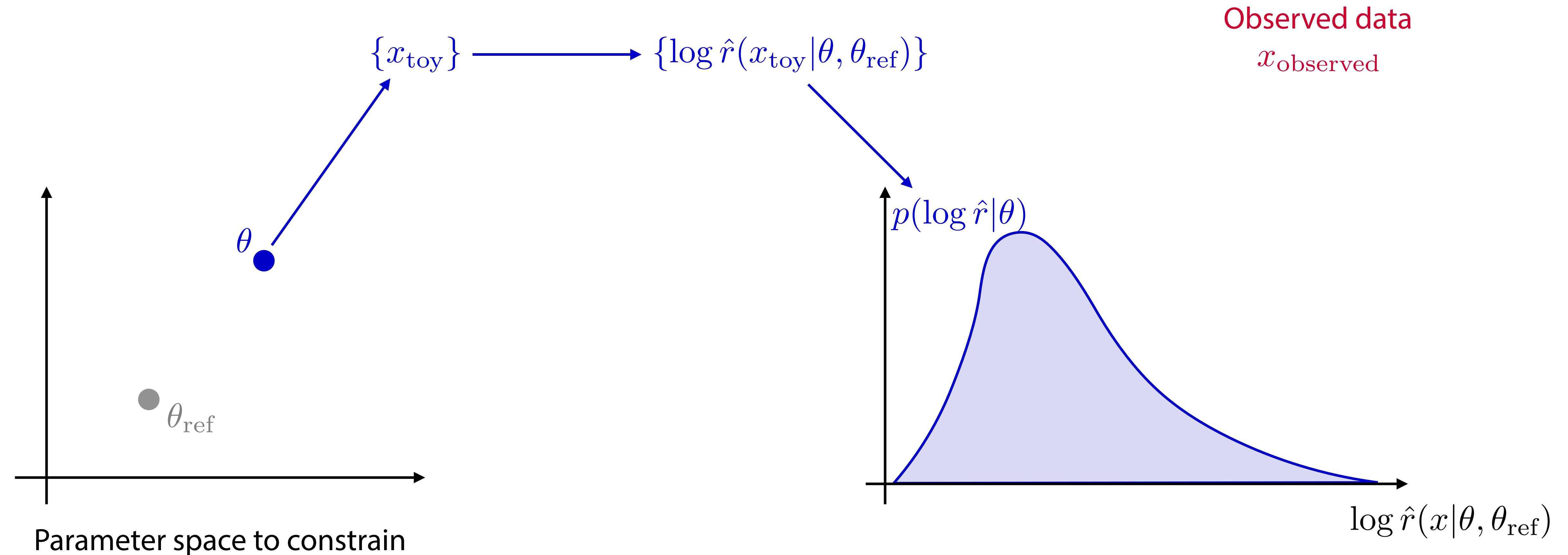


Frequentist inference

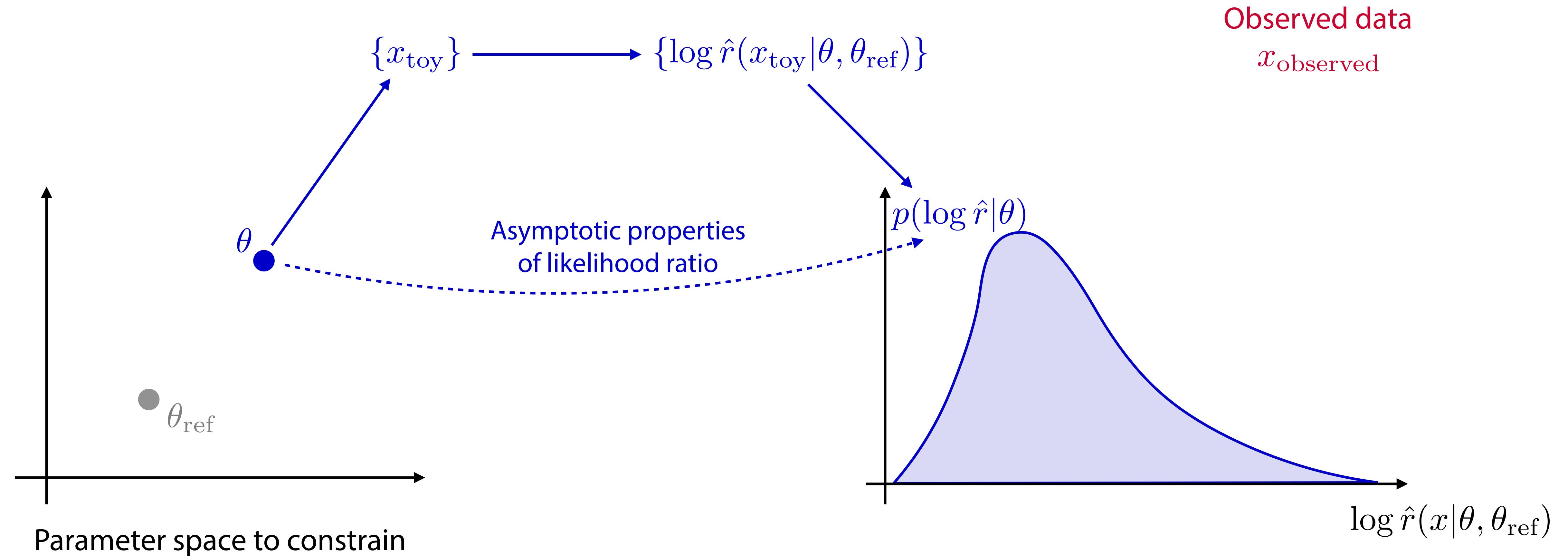


Observed data
 x_{observed}

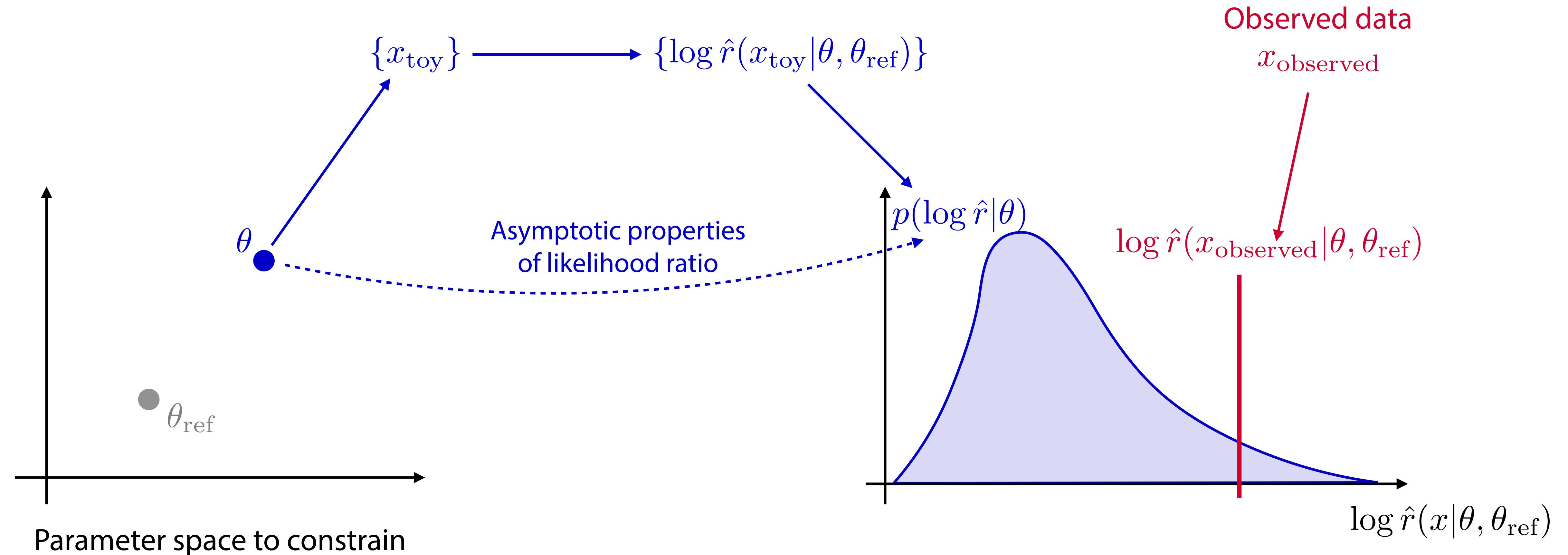
Frequentist inference



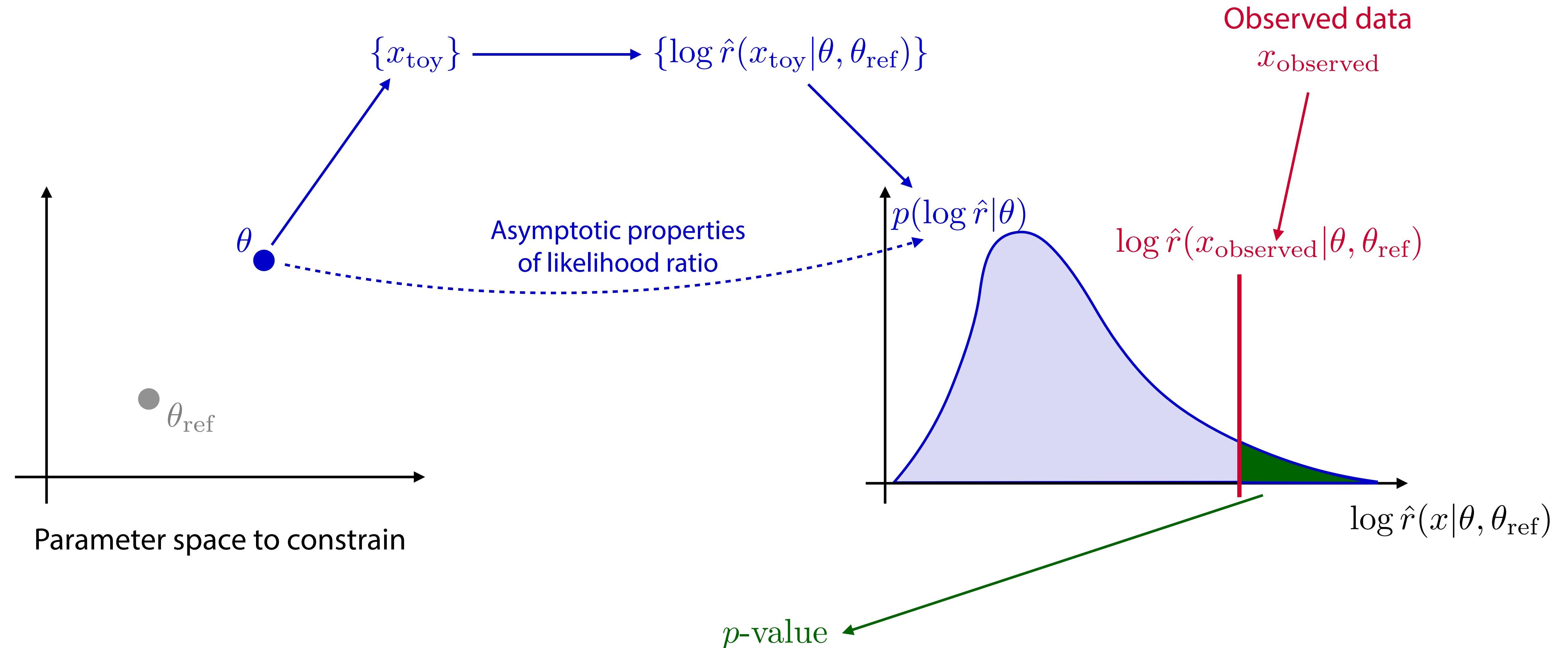
Frequentist inference



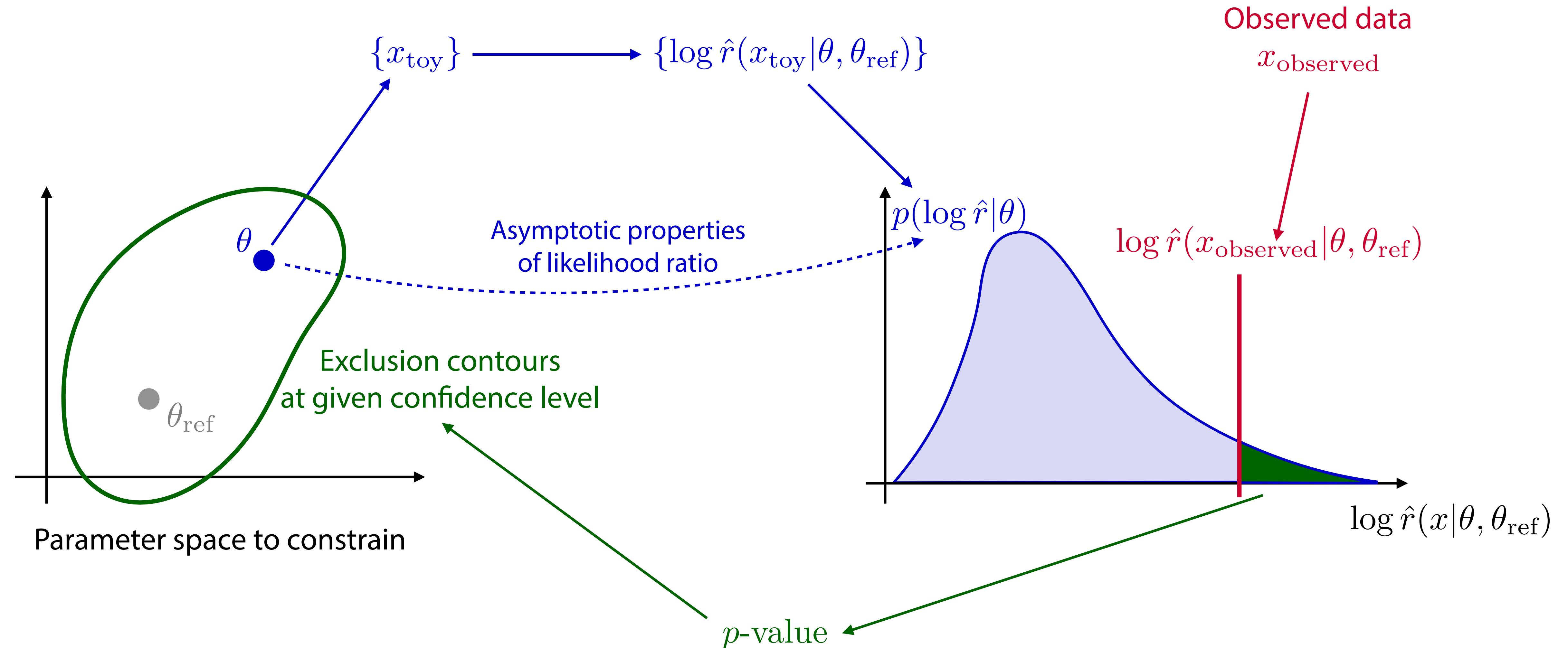
Frequentist inference



Frequentist inference



Frequentist inference

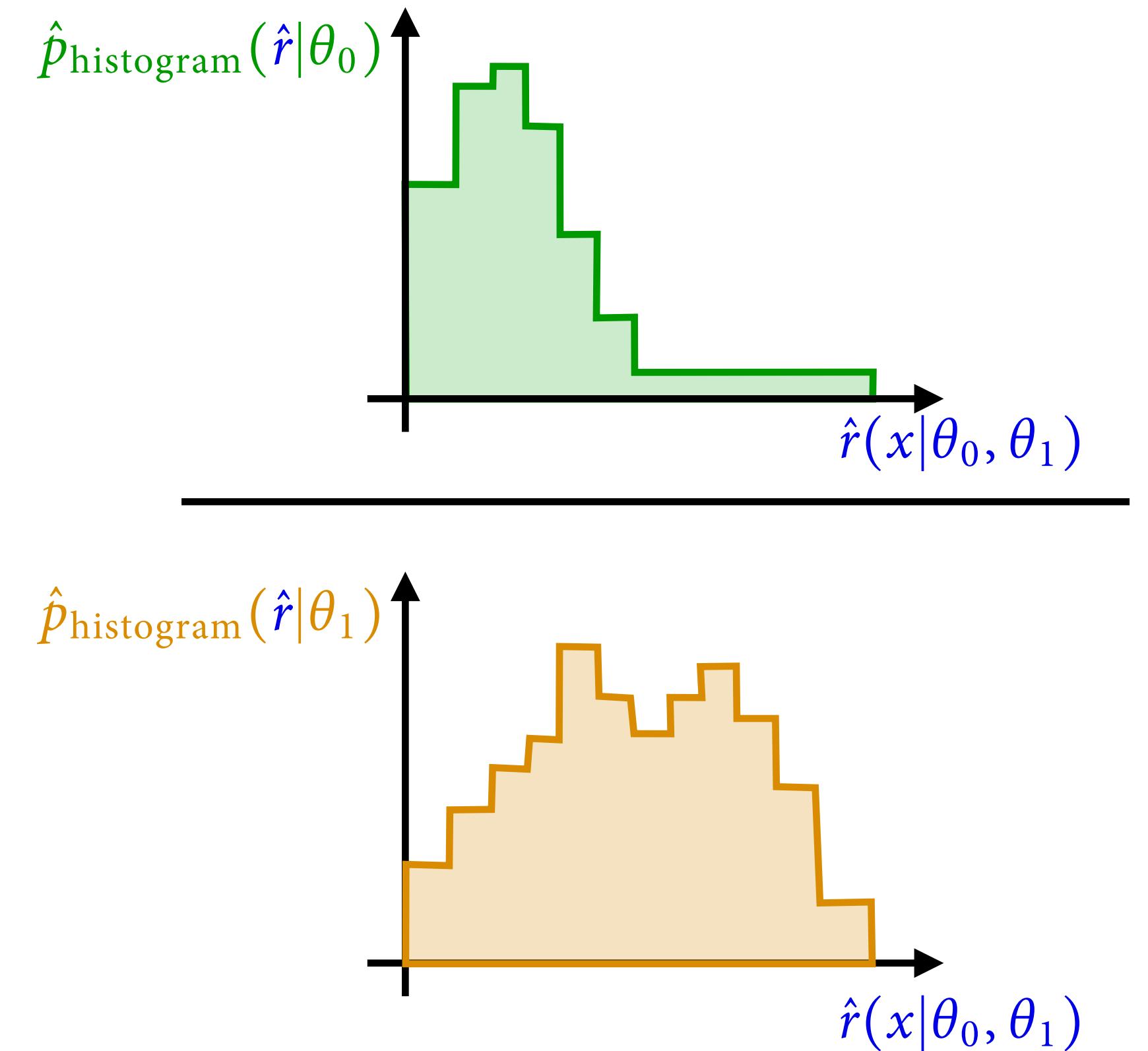


Calibration

[K. Cranmer J. Pavez, G. Louppe 1506.02169]

What if the NN likelihood ratio estimator $\hat{r}(x|\theta_0, \theta_1)$ is off? Calibrate!

$$\hat{r}_{\text{calibrated}}(x|\theta_0, \theta_1) = \frac{\hat{p}_{\text{histogram}}(\hat{r}(x|\theta_0, \theta_1)|\theta_0)}{\hat{p}_{\text{histogram}}(\hat{r}(x|\theta_0, \theta_1)|\theta_1)}$$



Inference results based on $\hat{r}_{\text{calibrated}}(x|\theta_0, \theta_1)$ have guaranteed coverage: they will not be too optimistic

Bonus material: particle physics

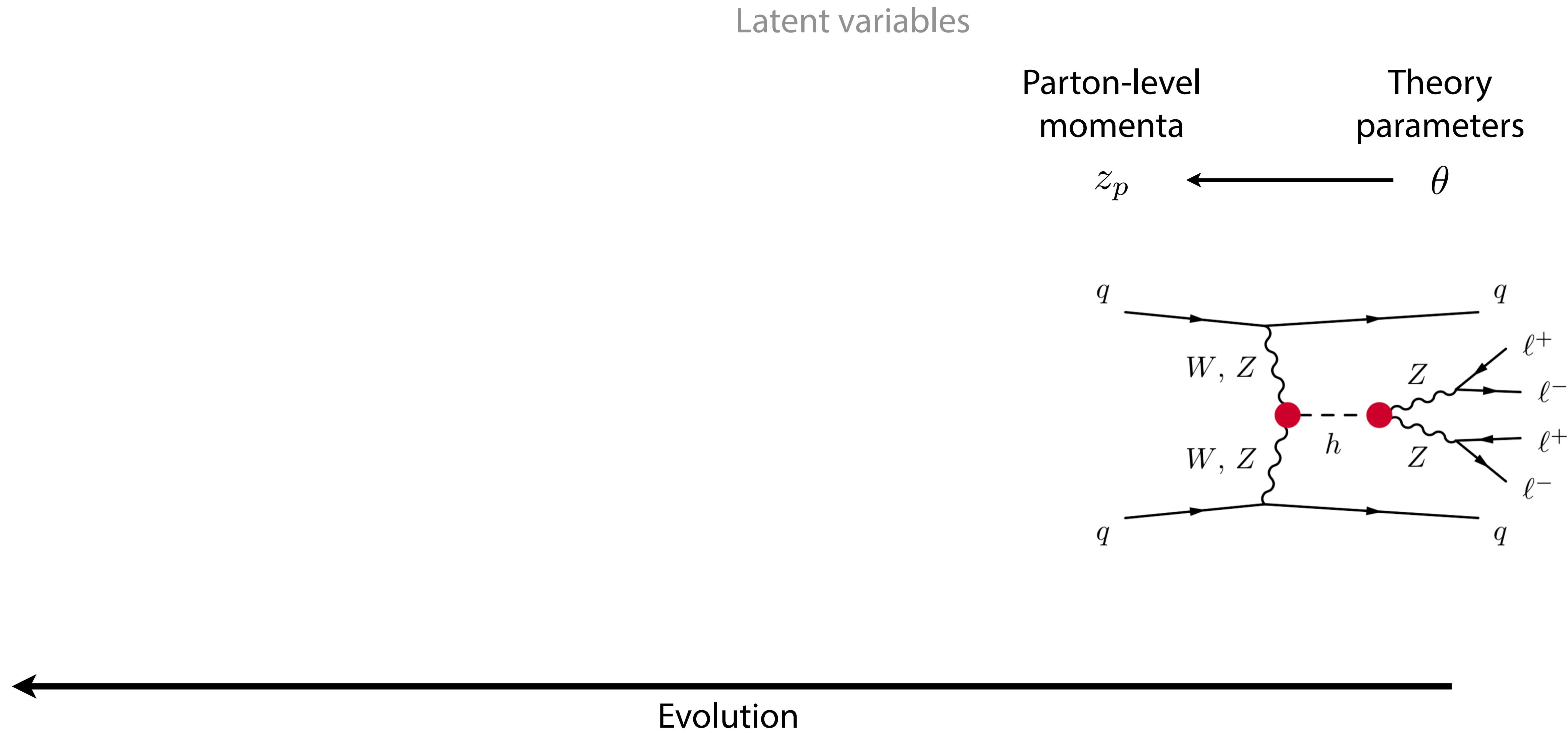
Modelling particle physics processes

Theory
parameters
 θ

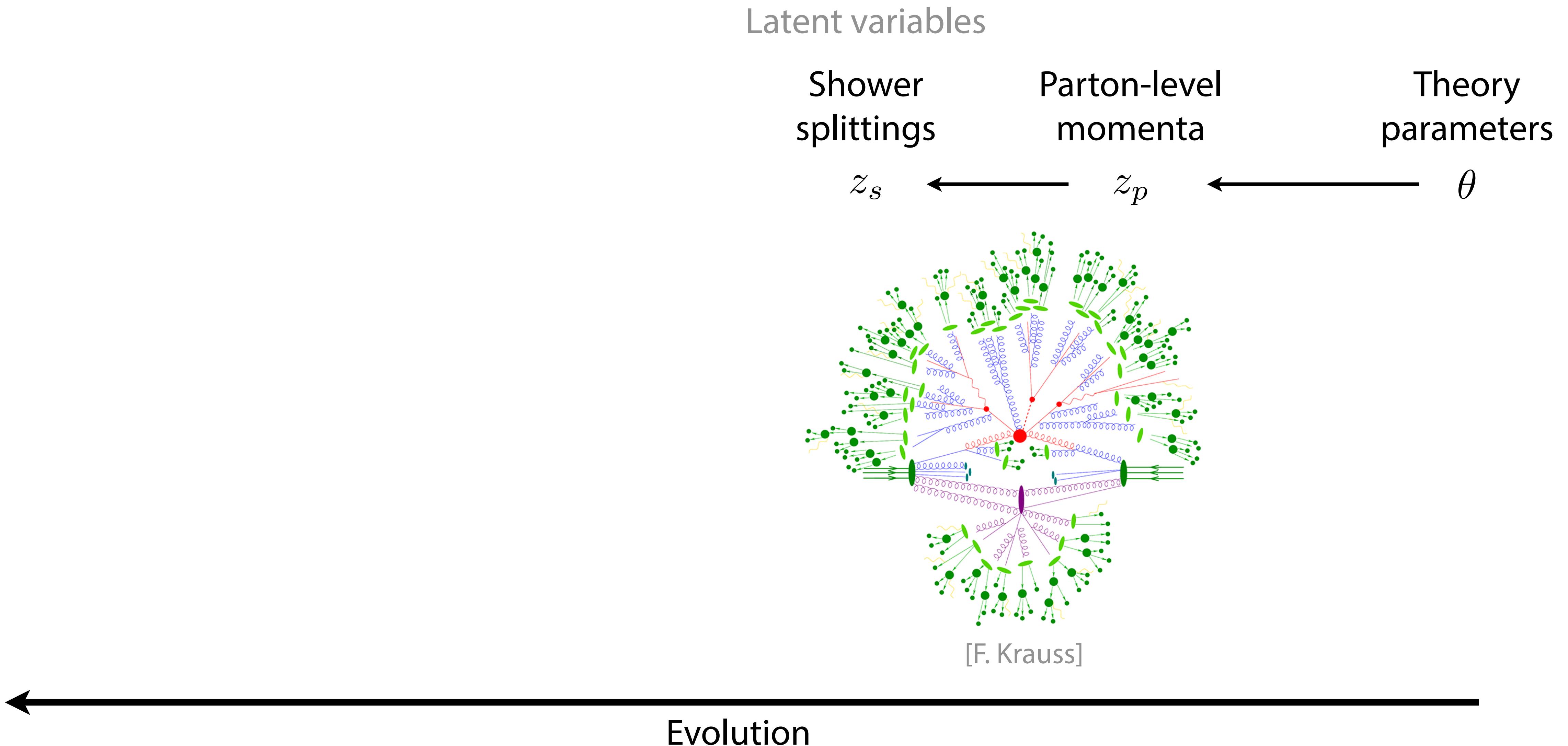


Evolution

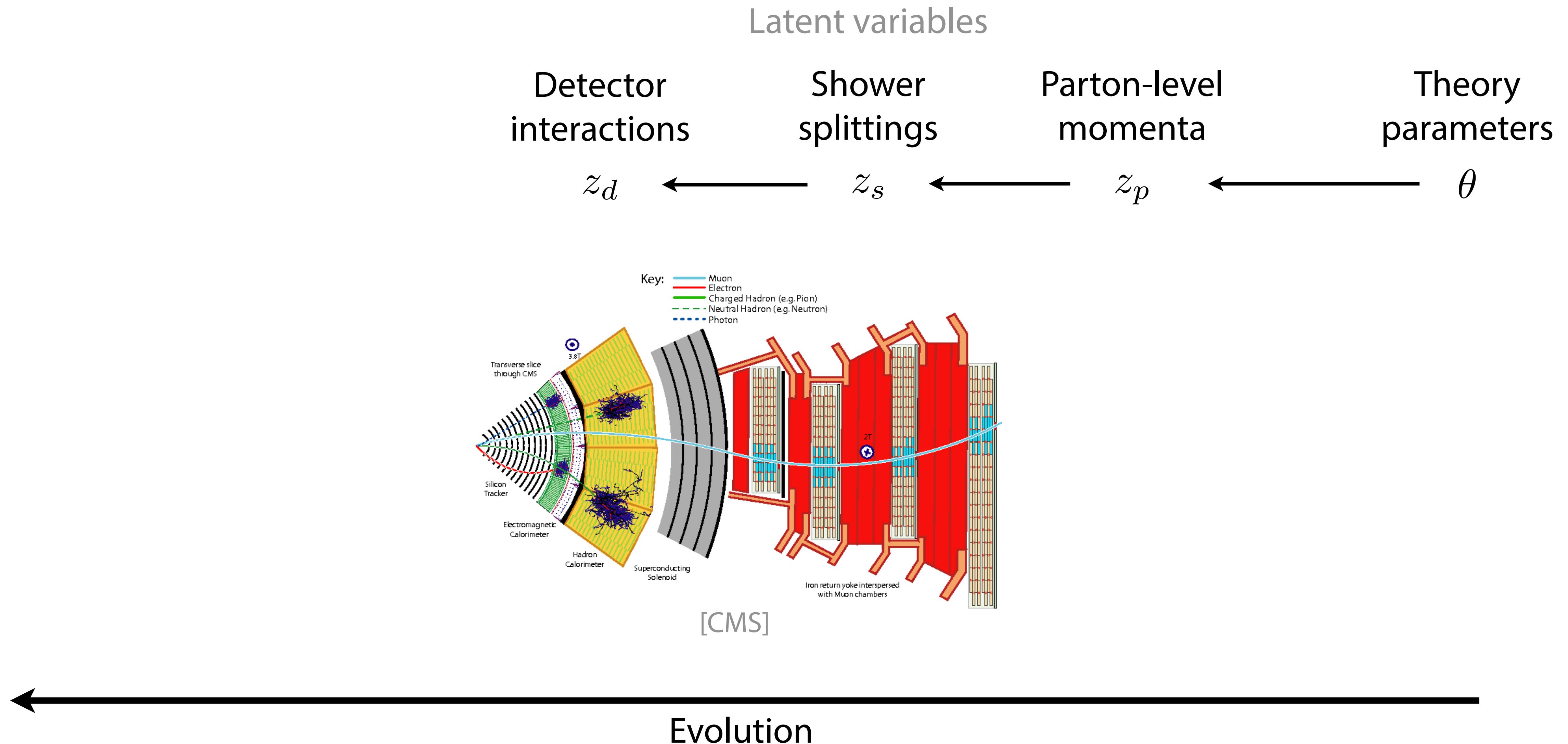
Modelling particle physics processes



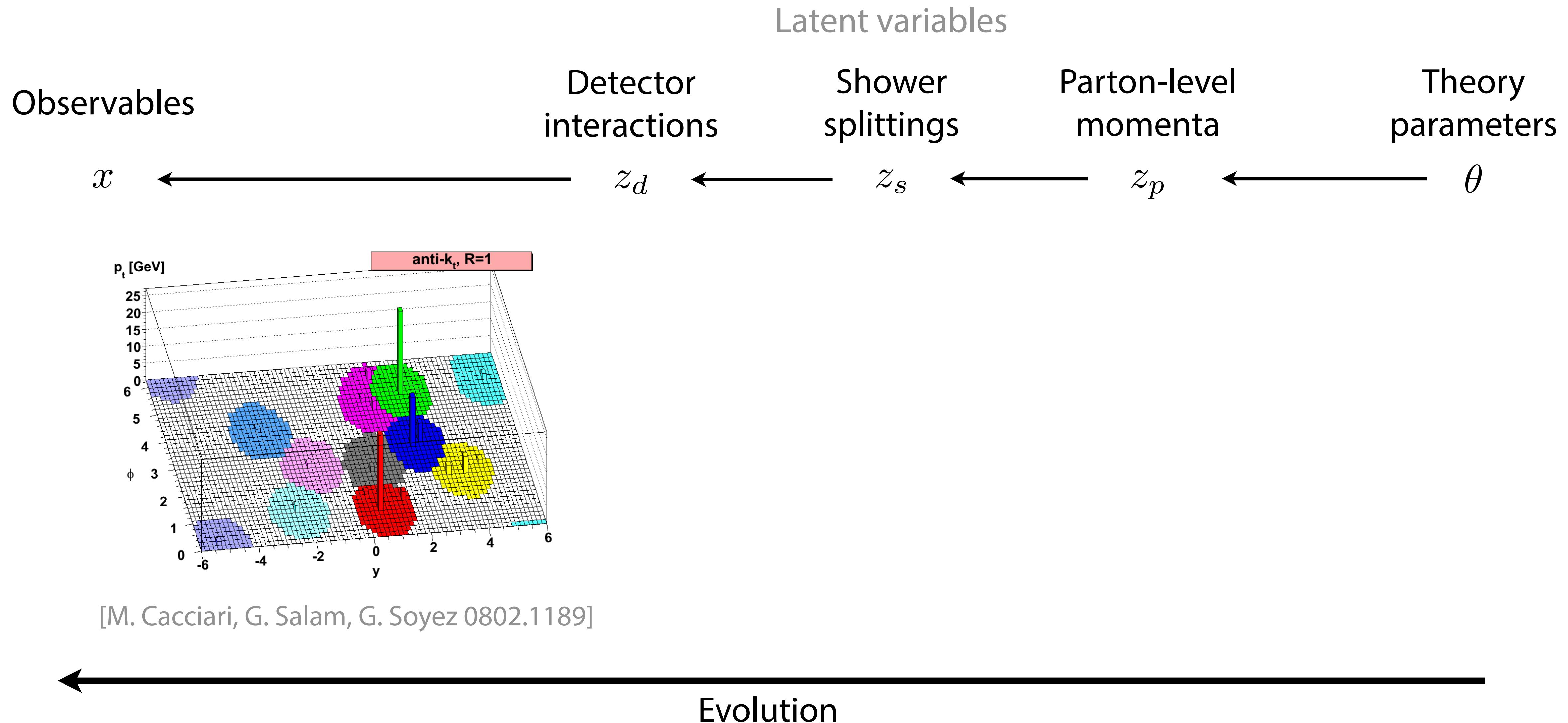
Modelling particle physics processes



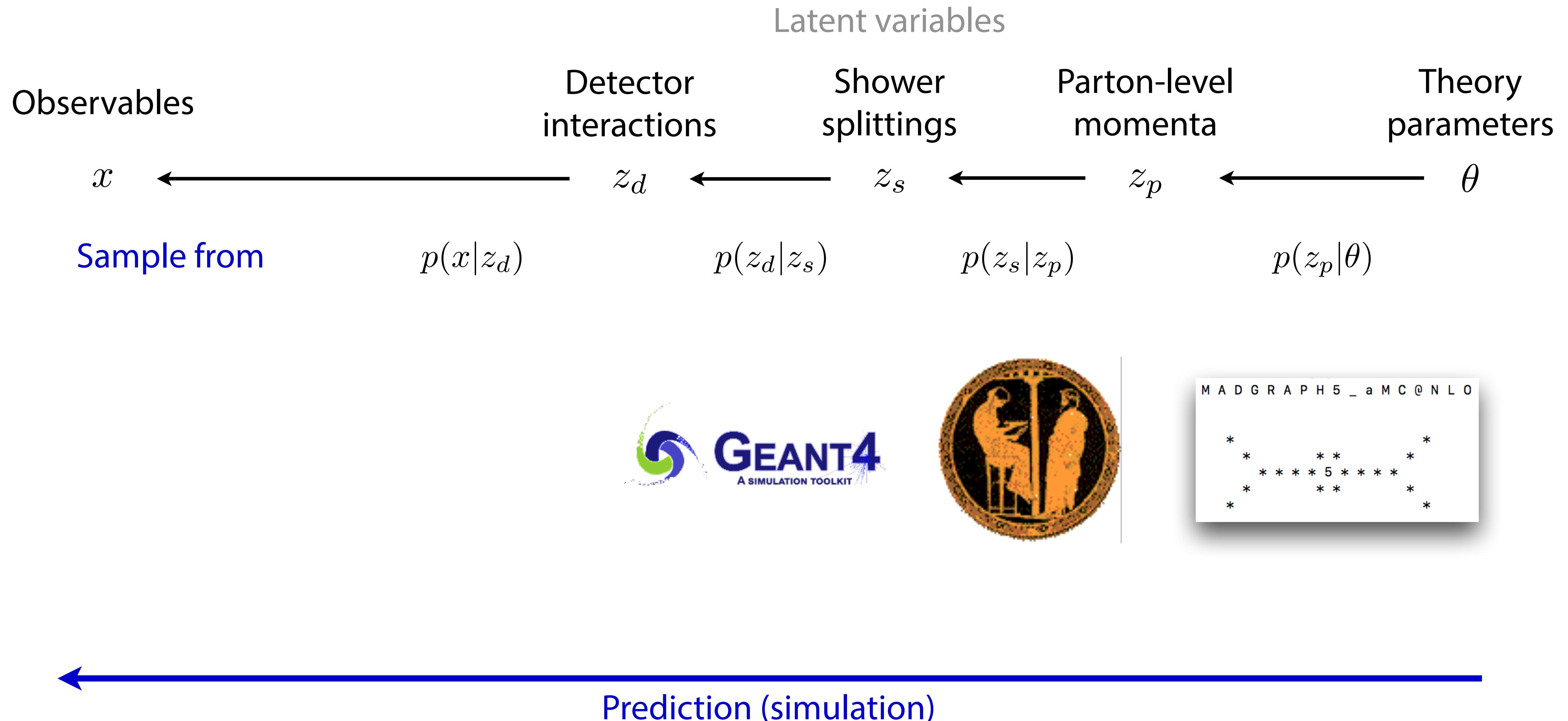
Modelling particle physics processes



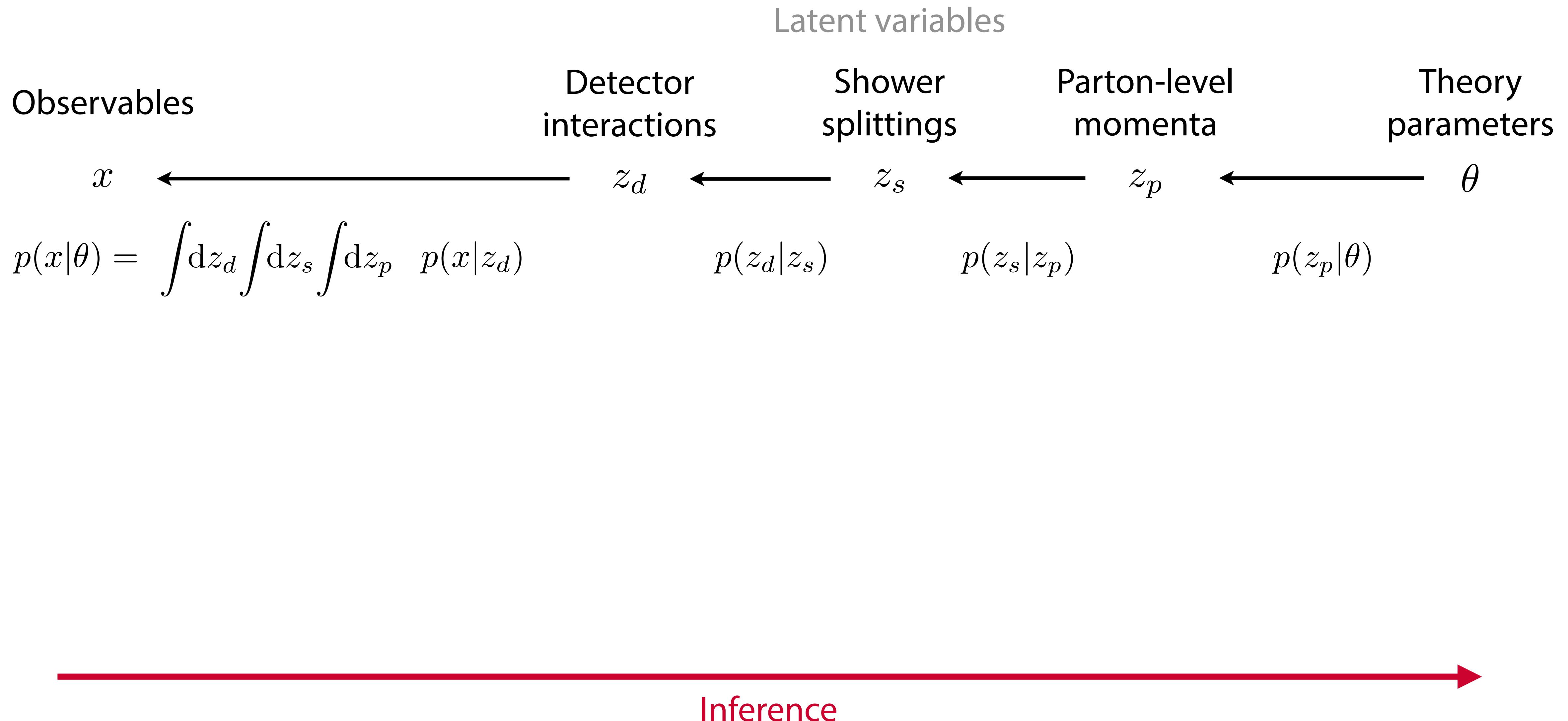
Modelling particle physics processes



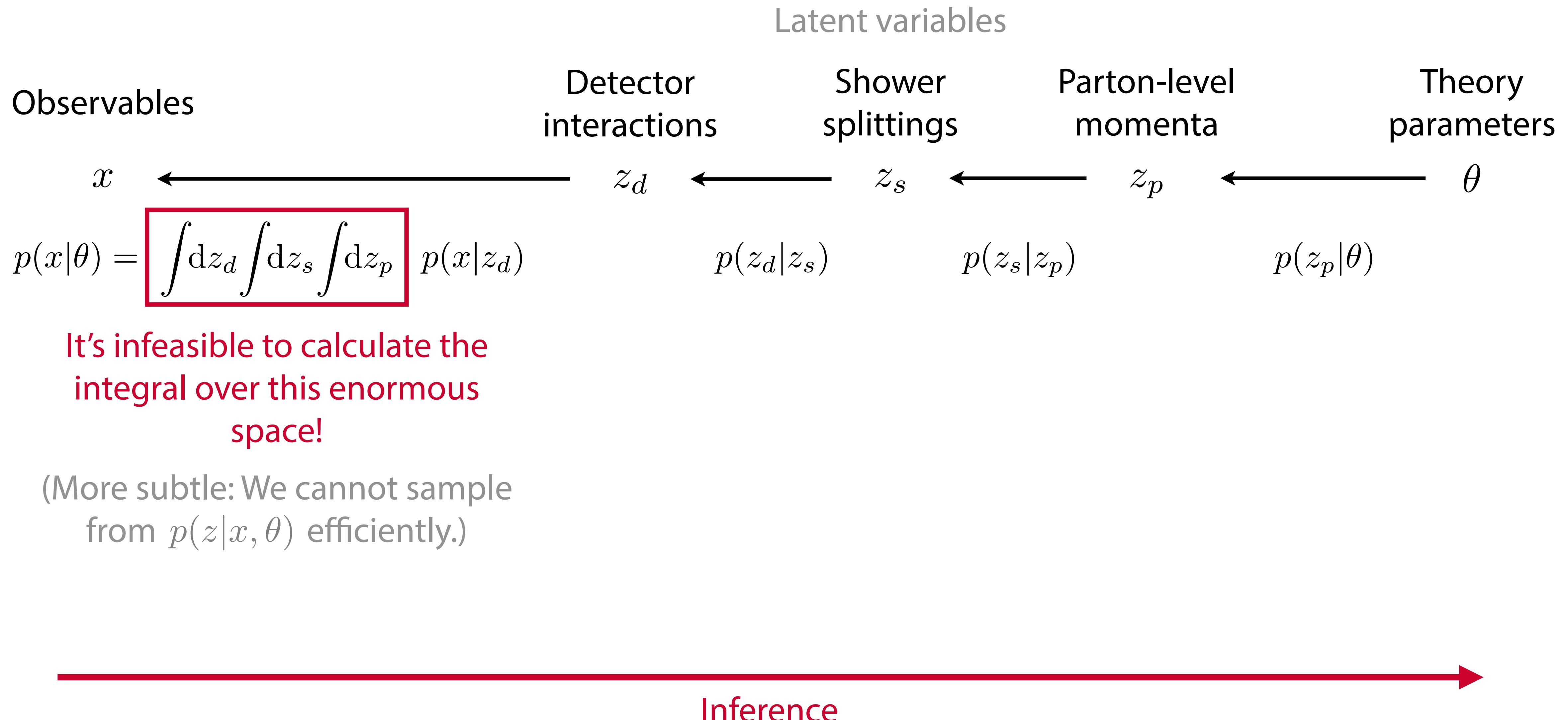
Modelling particle physics processes



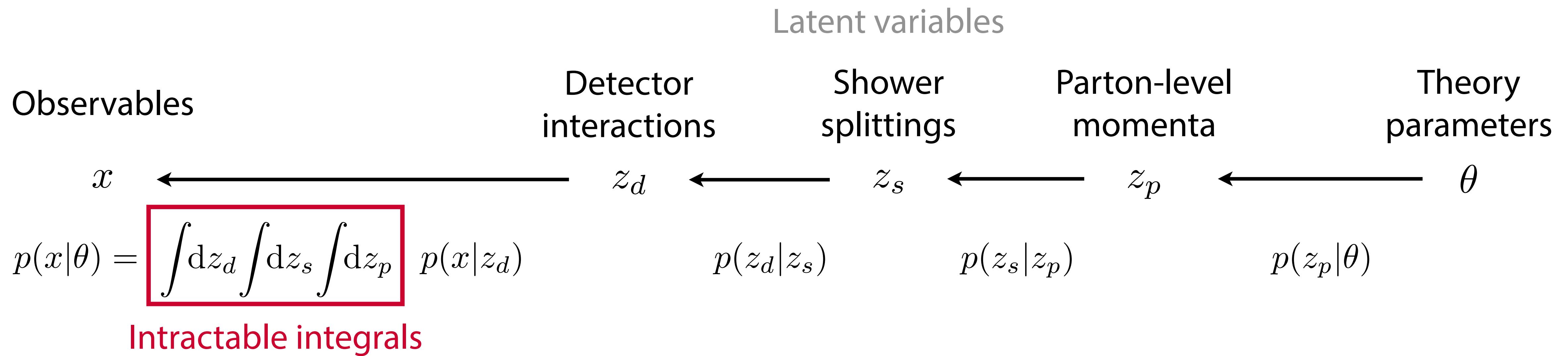
Modelling particle physics processes



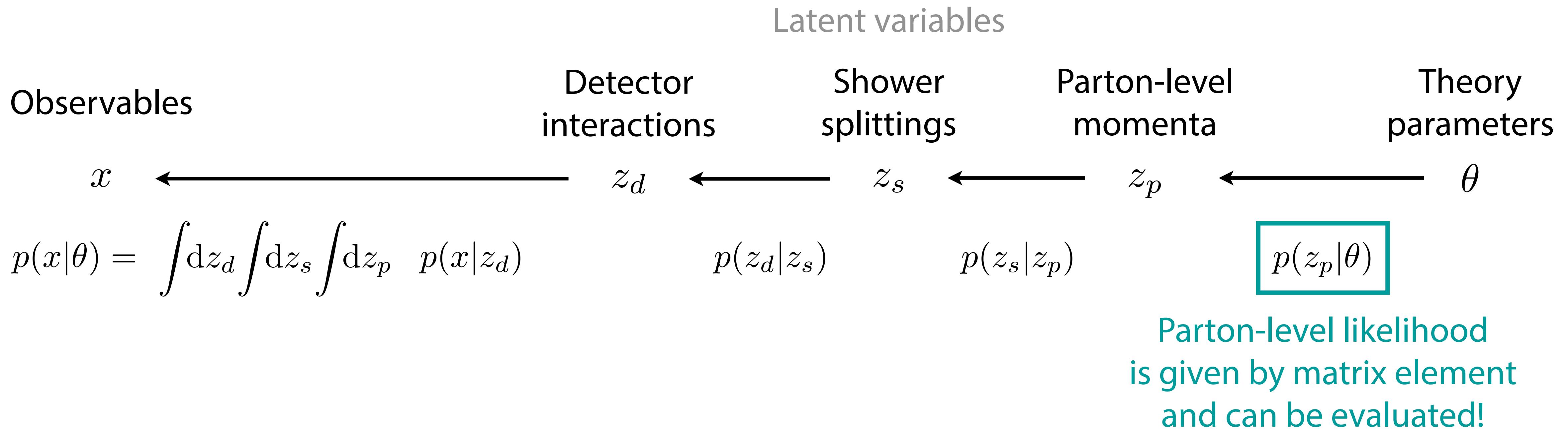
Modelling particle physics processes



Mining gold from the simulator



Mining gold from the simulator



⇒ For each simulated event, we can calculate the **joint likelihood ratio** which depends on the specific evolution of the simulation:

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

$$\frac{p(z_p|\theta_0)}{p(z_p|\theta_1)} \sim \frac{|\mathcal{M}(z_p|\theta_0)|^2}{|\mathcal{M}(z_p|\theta_1)|^2}$$

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

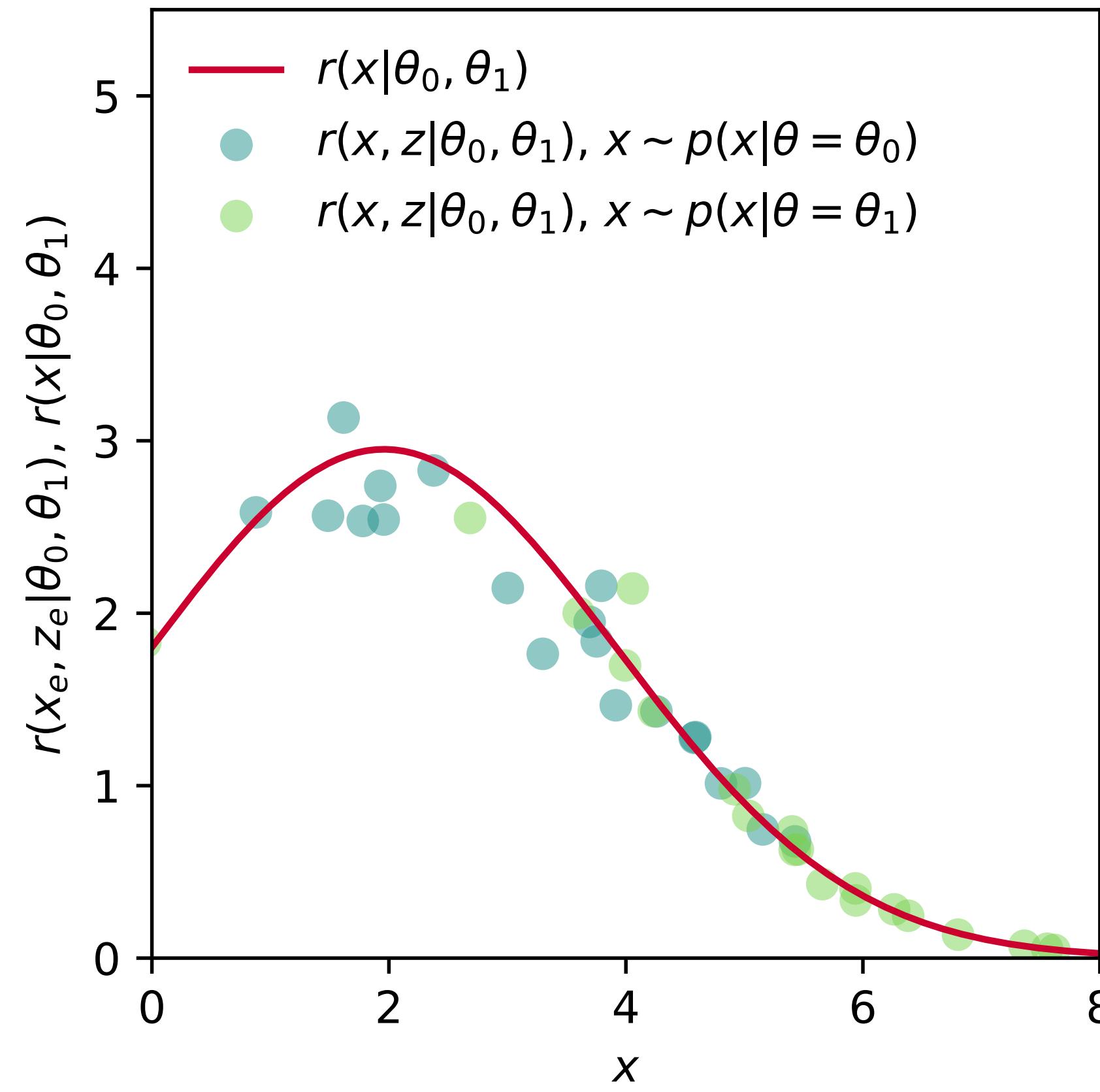
$$r(x, z | \theta_0, \theta_1) \equiv \frac{p(x, z_d, z_s, z_p | \theta_0)}{p(x, z_d, z_s, z_p | \theta_1)}$$



$r(x, z | \theta_0, \theta_1)$ 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

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

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

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

It is minimized by

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

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

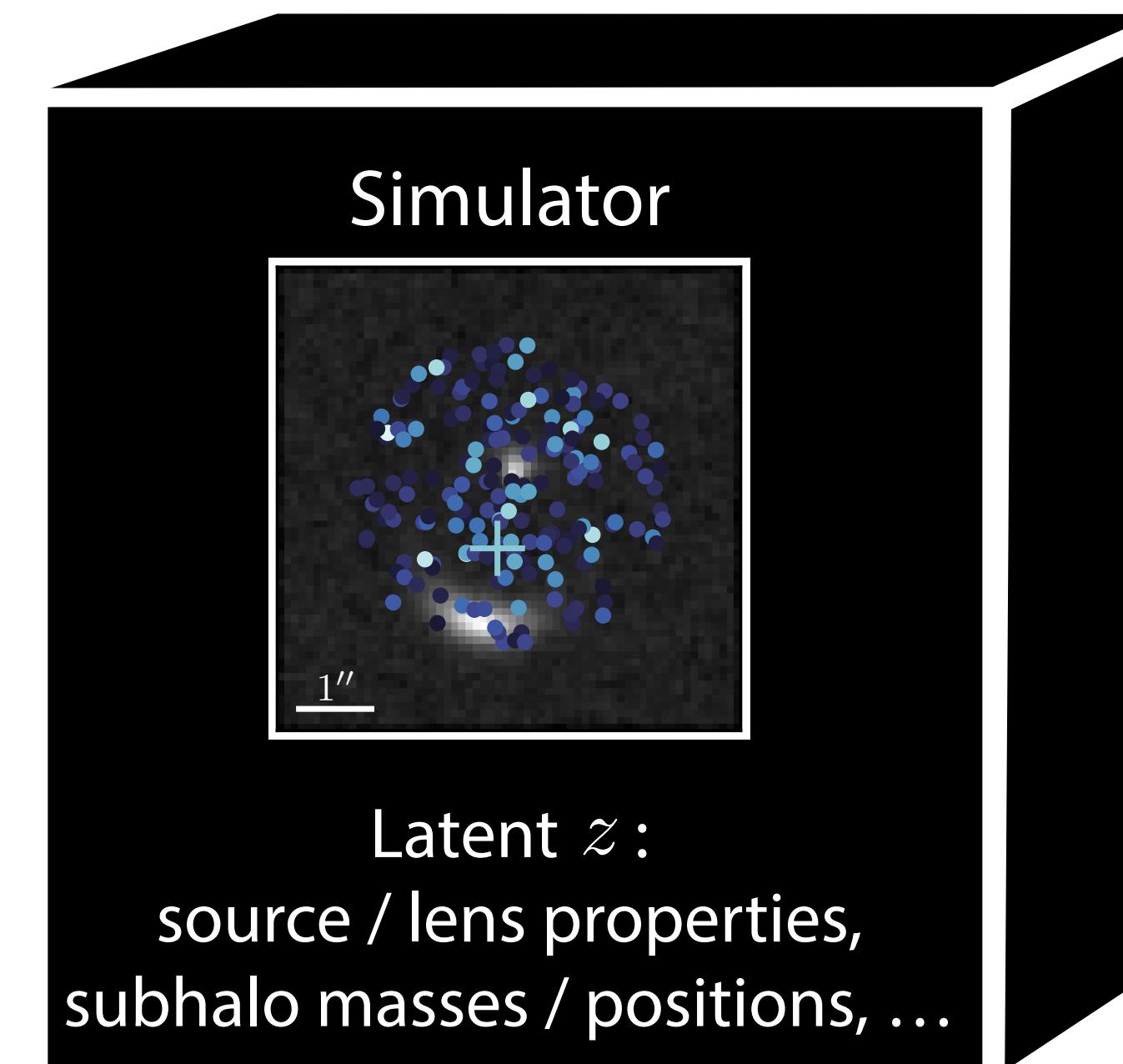
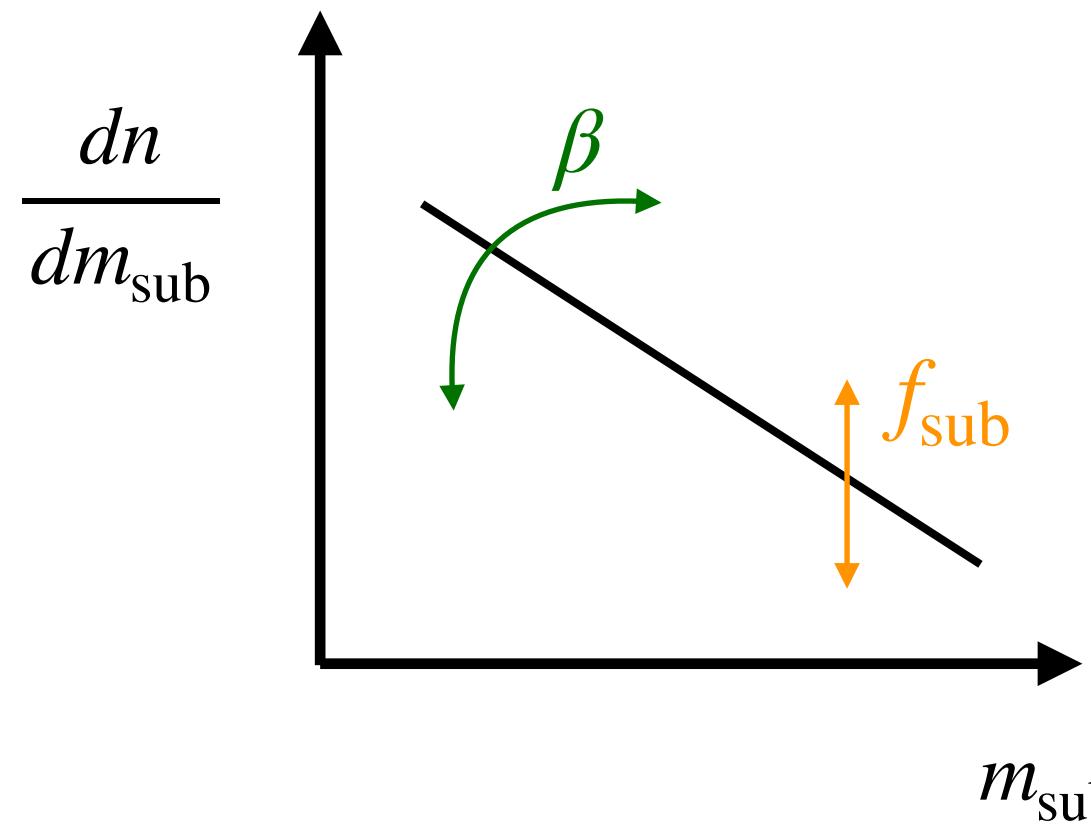
We want the likelihood ratio function

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

Bonus material: gravitational lensing

Overview

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



64² observables x

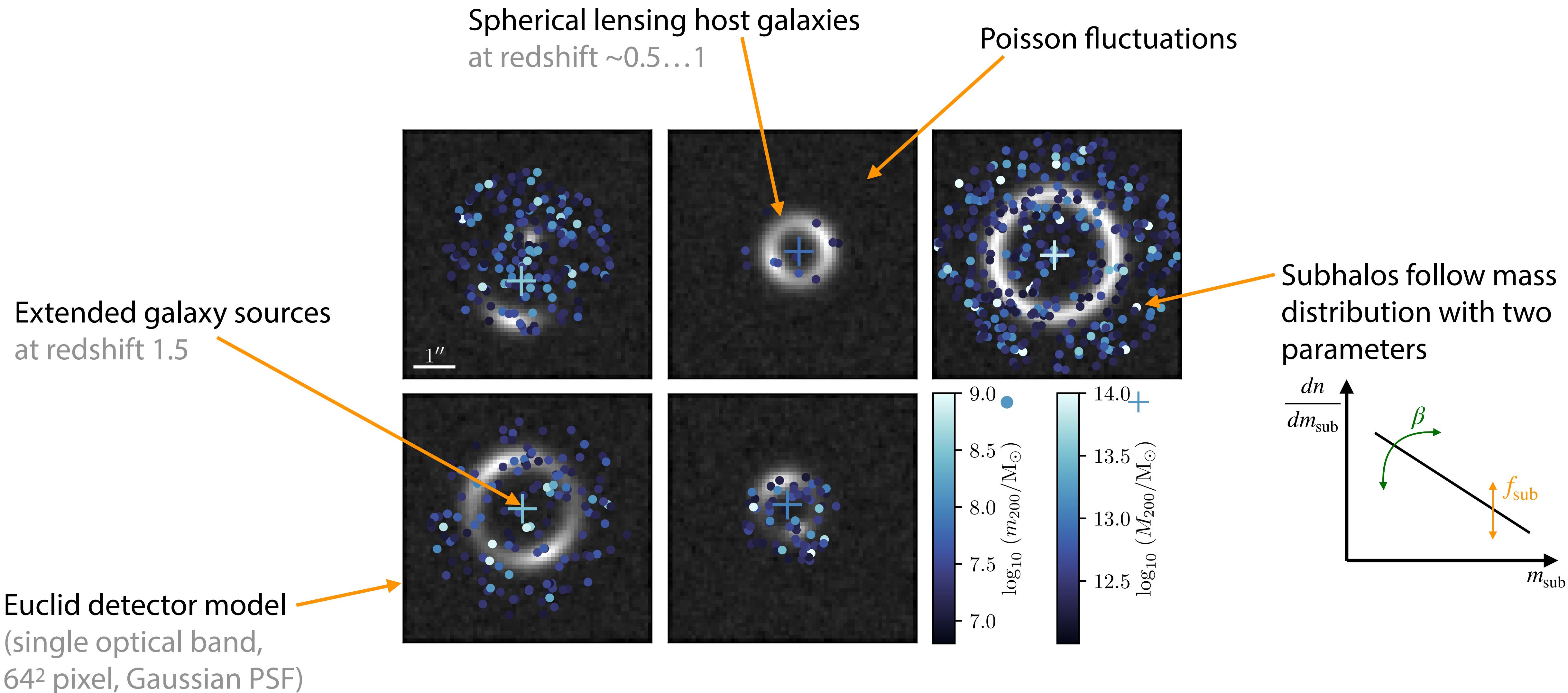


Prediction: We construct a simulator that can sample $x \sim p(x|\theta)$

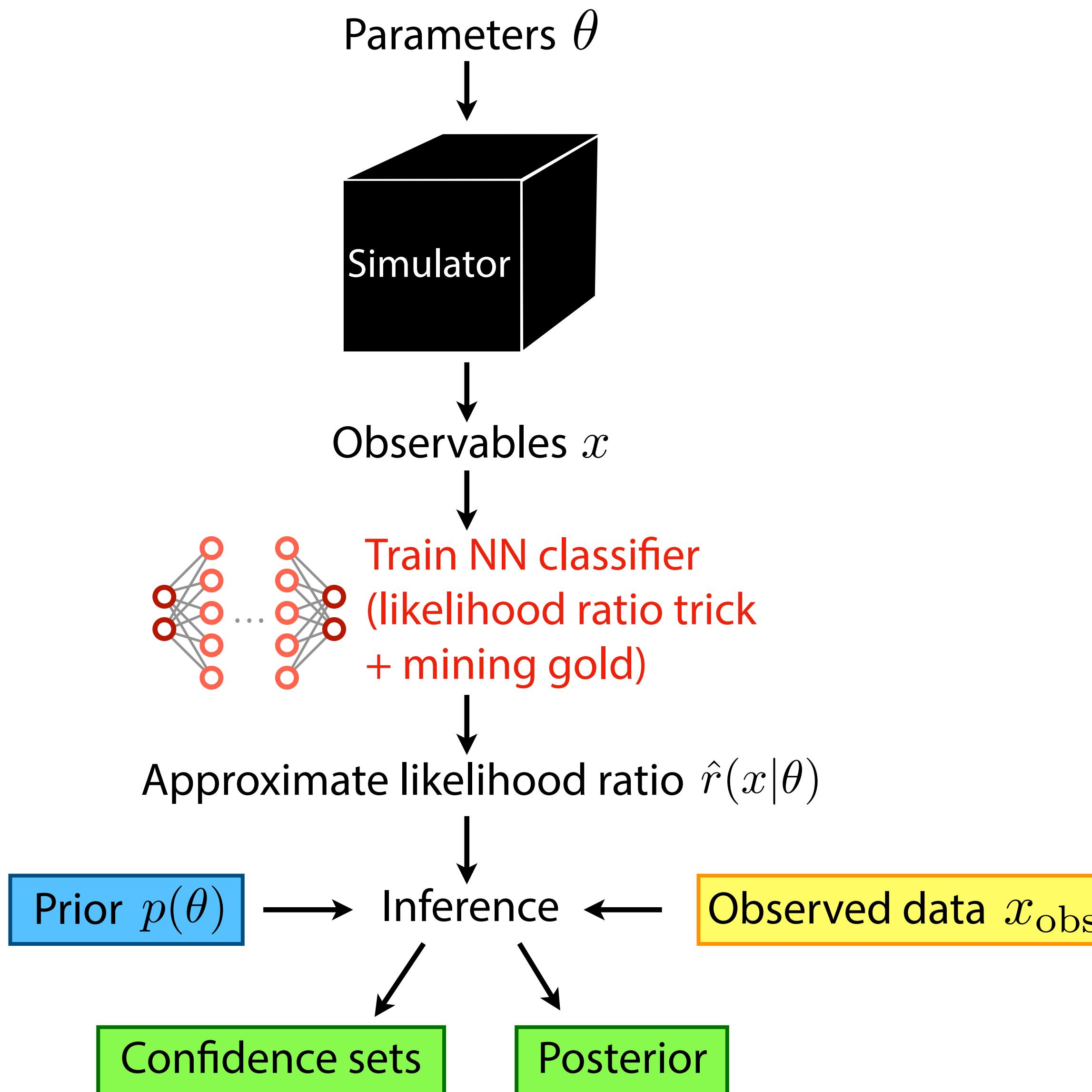
Inference: We train neural likelihood ratio estimators $\hat{r}(x|\theta)$

Proof-of-principle simulator

[following T. Collett 1507.02657]



Inference setup



Training data: 10^6 lensed images with
 $0 \leq f_{\text{sub}} \leq 0.2, -1.5 \leq \beta \leq -0.5$

Convolutional neural network (modified ResNet-18)
trained on ALICES loss
[M. Stoye, JB, J. Pavez, G, Louppe, K. Cranmer 1808.00973]

Calibration of network output

Synthetic “observed” data set: $f_{\text{sub}} = 0.05, \beta = -0.9$

Bayesian & frequentist inference

LHC footnotes

- Full LHC likelihood:

$$p_{\text{full}}(\{x\}|\theta) = \text{Pois}(n|L\sigma(\theta)) \prod_{\text{events } x} p(x|\theta)$$

Total rate term:

- How likely is it to observe n events after cuts?
- “Easy” to compute
- For simplicity, we ignore this part in this talk

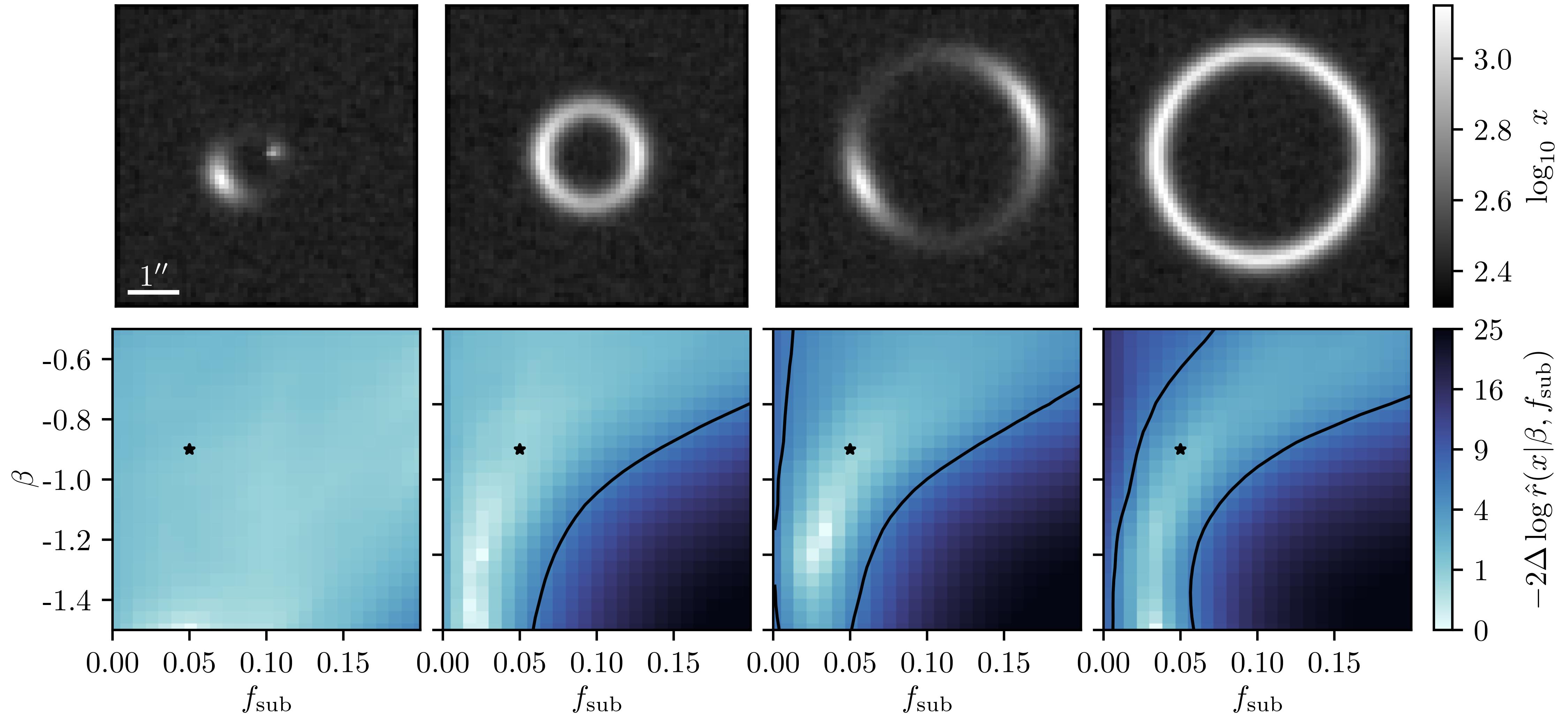
Kinematic term for each event:

- How likely is it that an event looks like it does?
- \sim normalized differential xsec
- This is the intractable part of the likelihood
- Focus of this talk

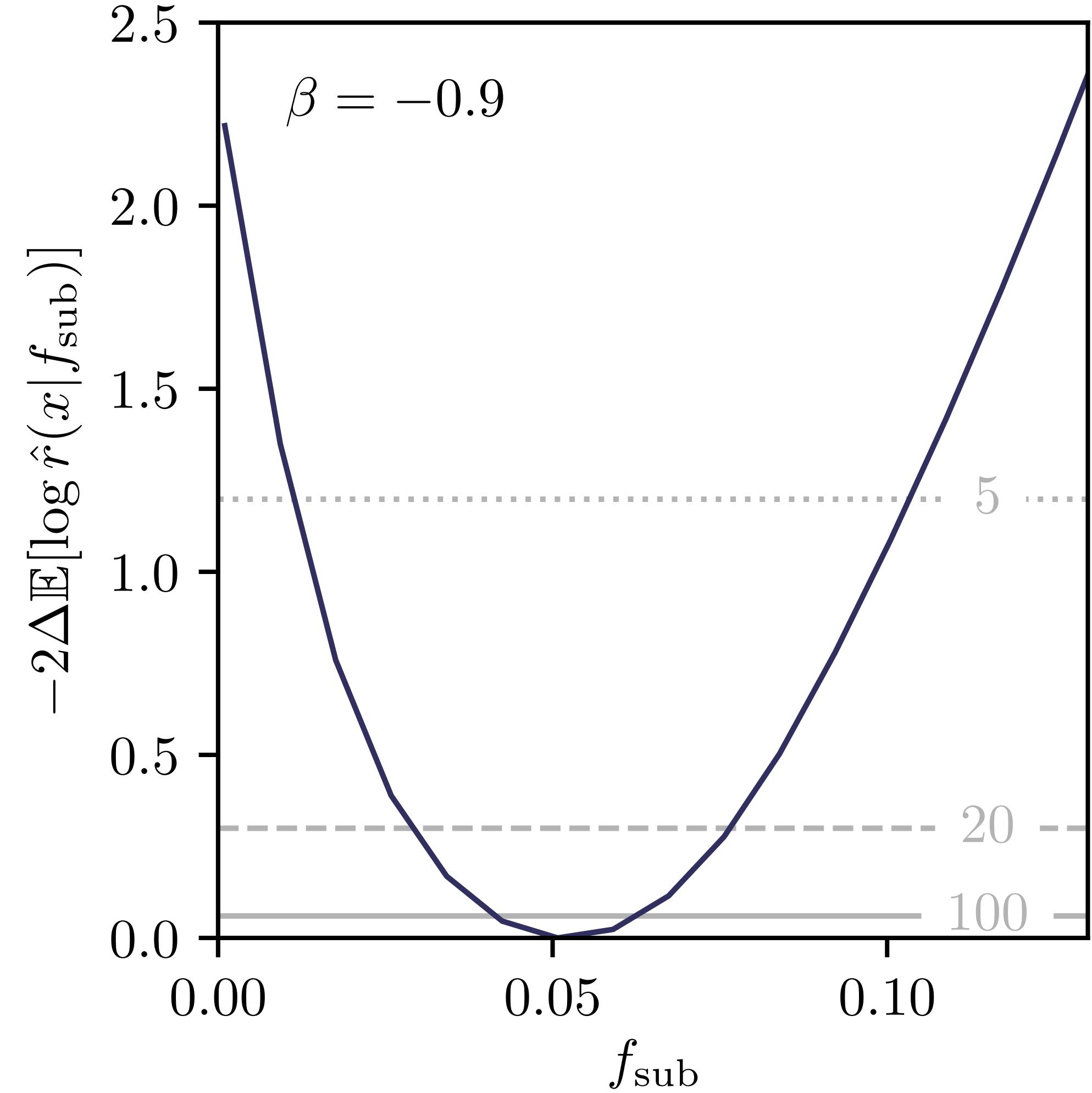
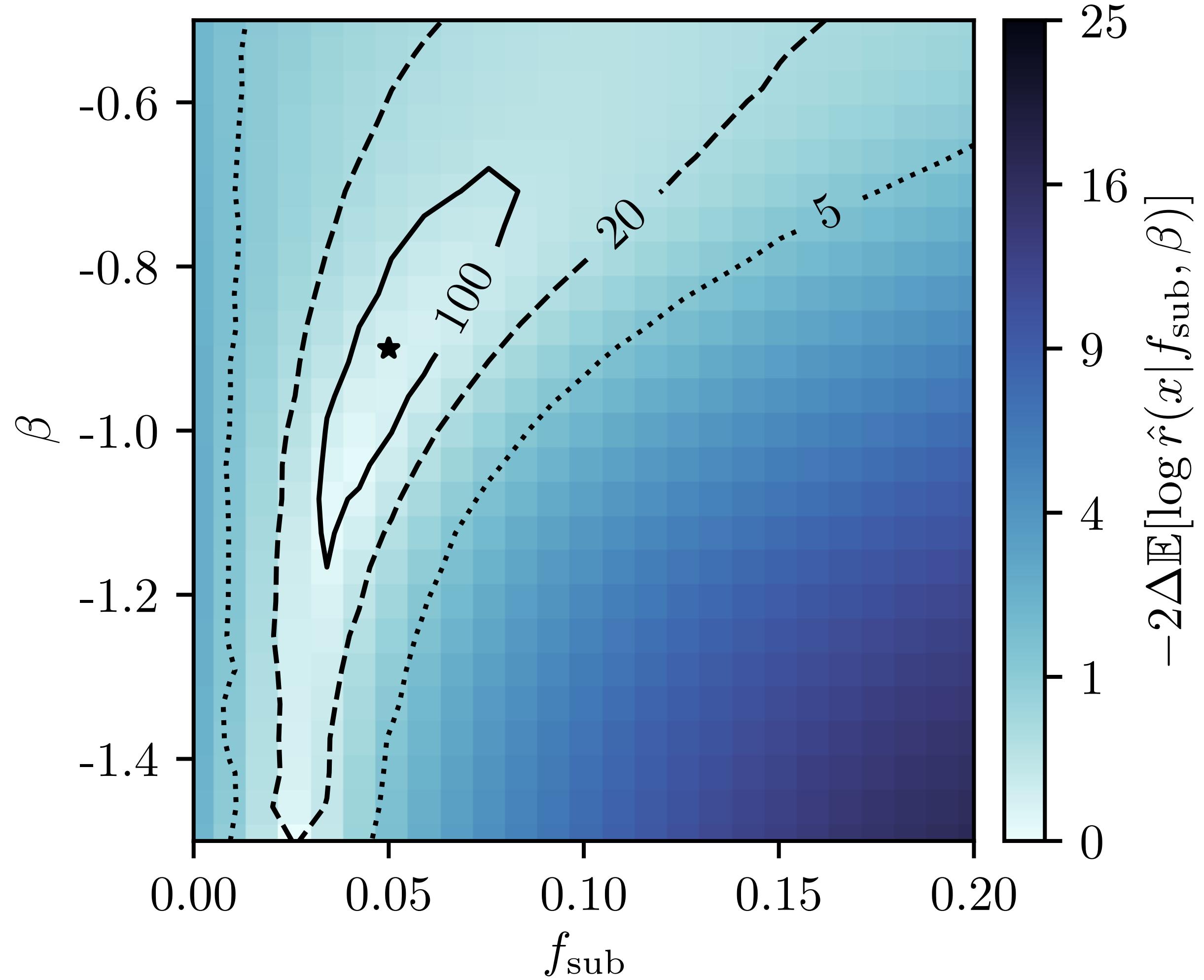
- Event selection:

- Choice of cuts shifts information between rate and kinematic part
- “Good” cuts depend on inference strategy
- This talk: assume fixed event selection

Inferring parameters from individual images

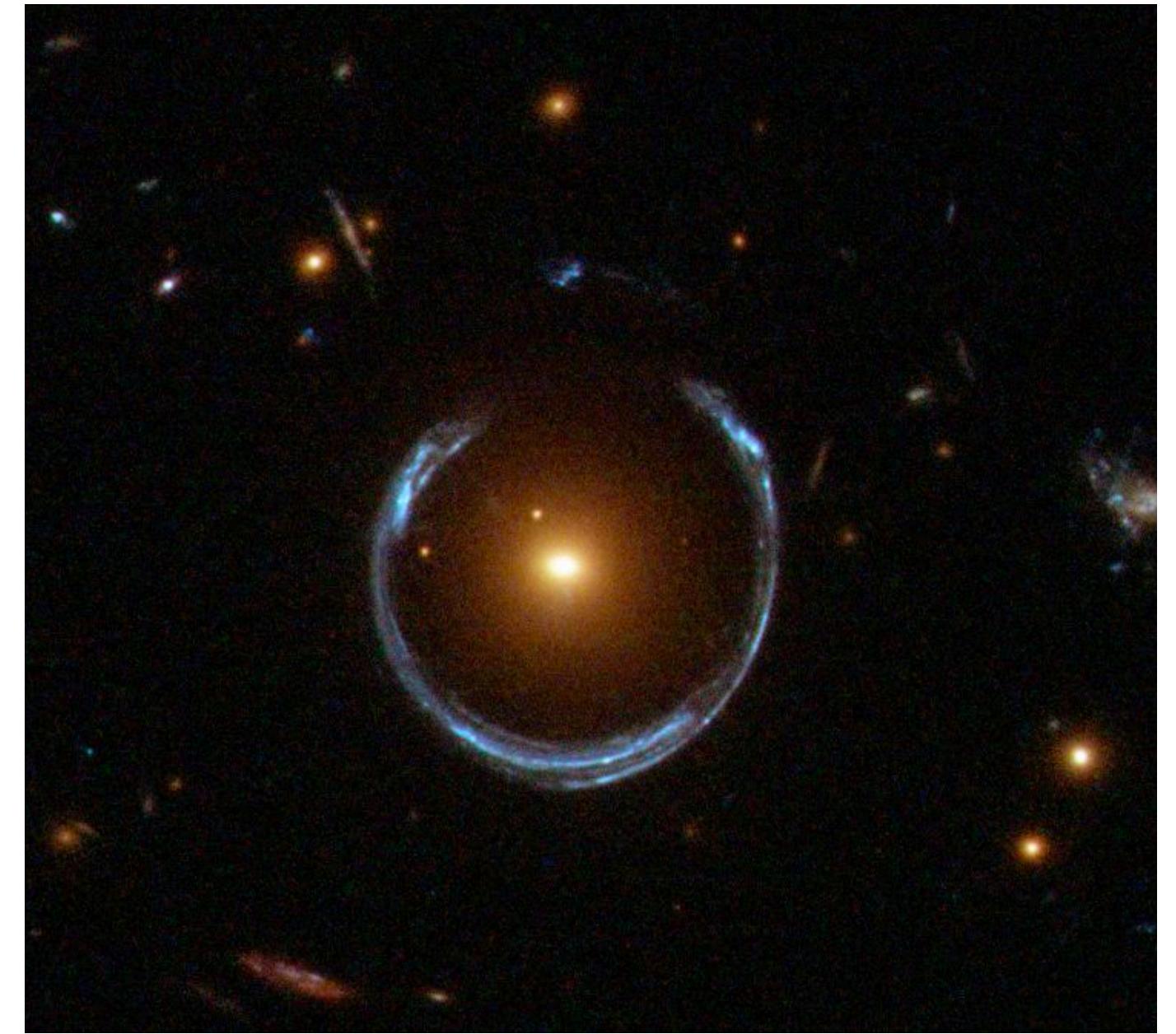


Expected likelihood ratio map



All the things we didn't do

- More involved subhalo mass function
 - Warm DM with DM mass as parameter
- Realistic simulators
 - More diverse source and host galaxies (e.g. data-driven)
 - Realistic subhalo modelling (tidal disruption, redshift dependence...)
 - Line-of-sight substructure
 - Realistic observation model (variable exposure / PSF, multiple bands...)
- Use auxiliary information during inference
- Evaluation on real data



[ESA/Hubble/NASA]

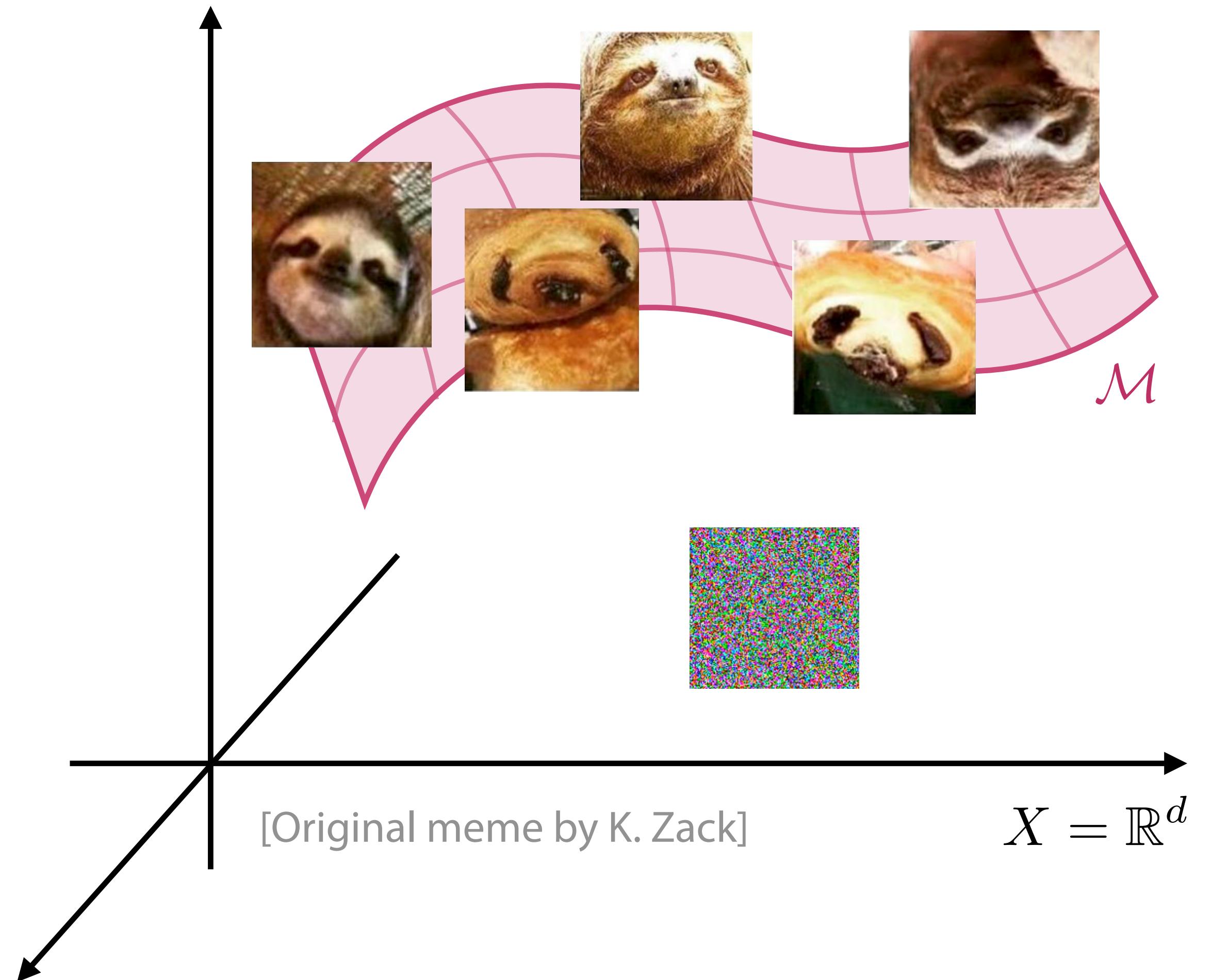
⇒ Our method should scale to a realistic setting, but will require more simulations and careful sanity checks

Bonus material: \mathcal{M} -flows

The manifold hypothesis

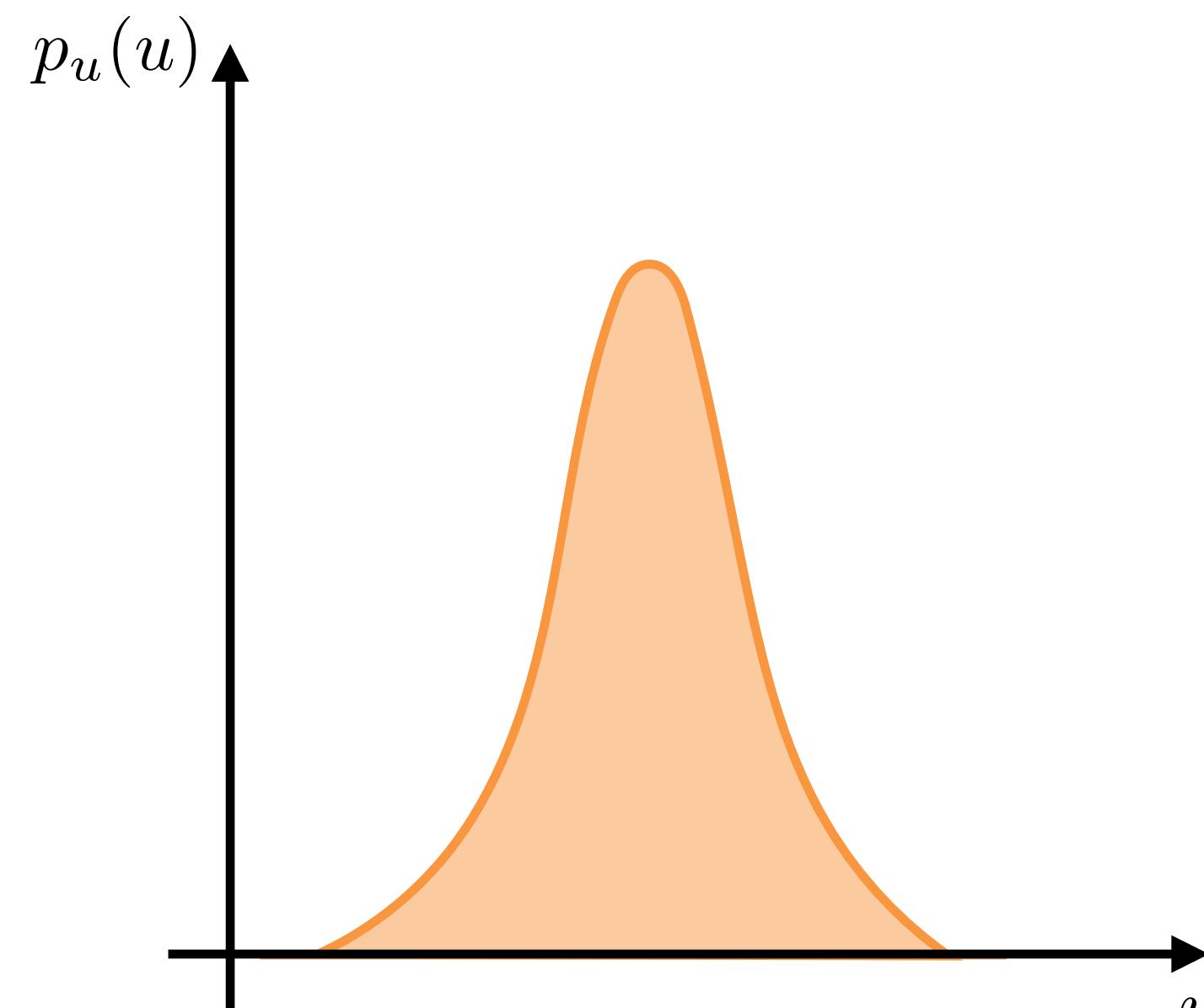
Data often live on a *n*-dimensional manifold embedded in the *d*-dimensional ambient space

- Robot arms, molecules: limited degrees of freedom
- Particle physics: energy-momentum conservation, on-shell conditions, redundant observables
- Many other high-dimensional datasets (e.g. images): empirical evidence for (approximate) data manifold [L. Cayton 2005; ...]



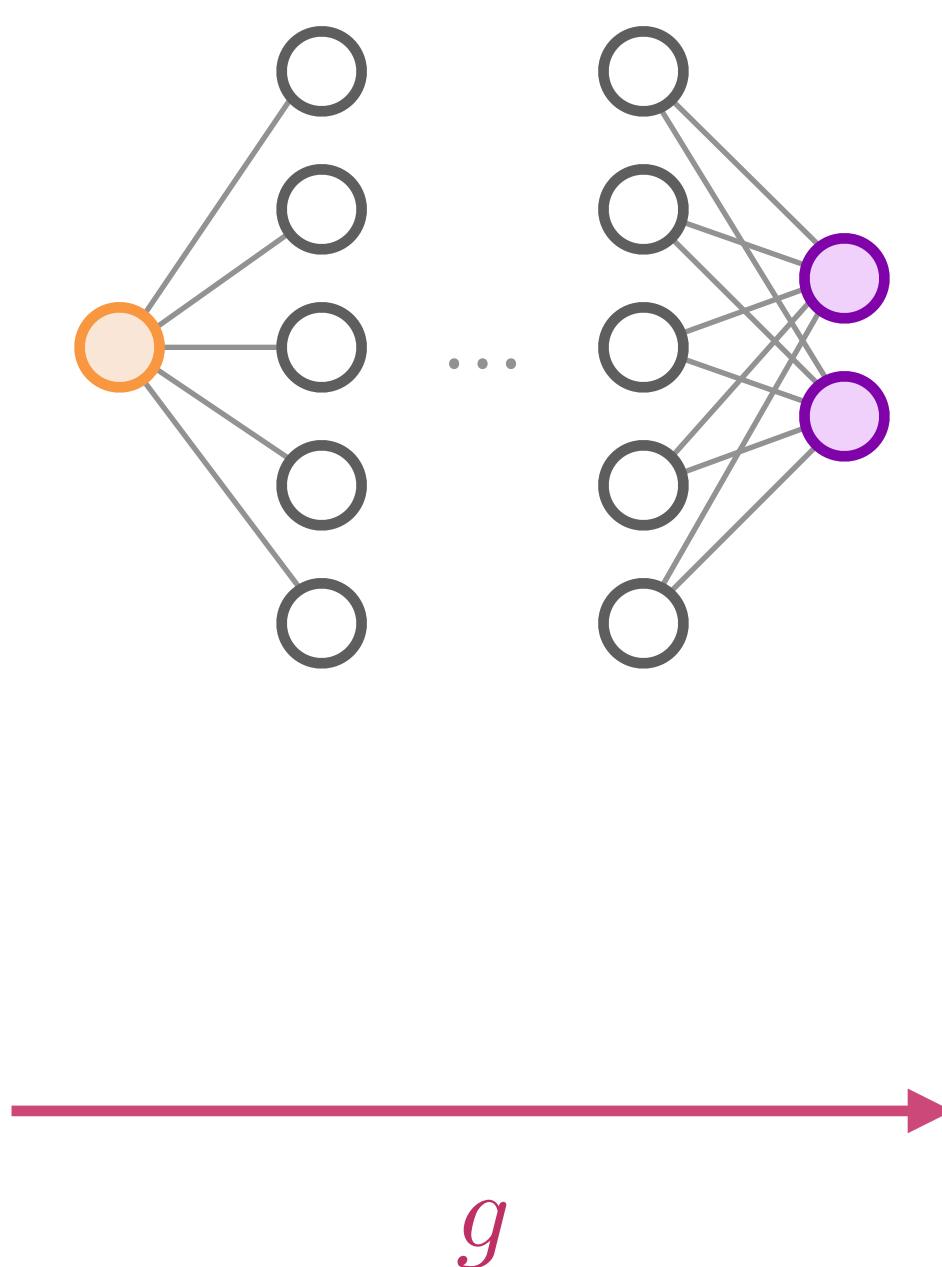
Generative adversarial networks (GANs)

[I. Goodfellow et al 1406.2661]

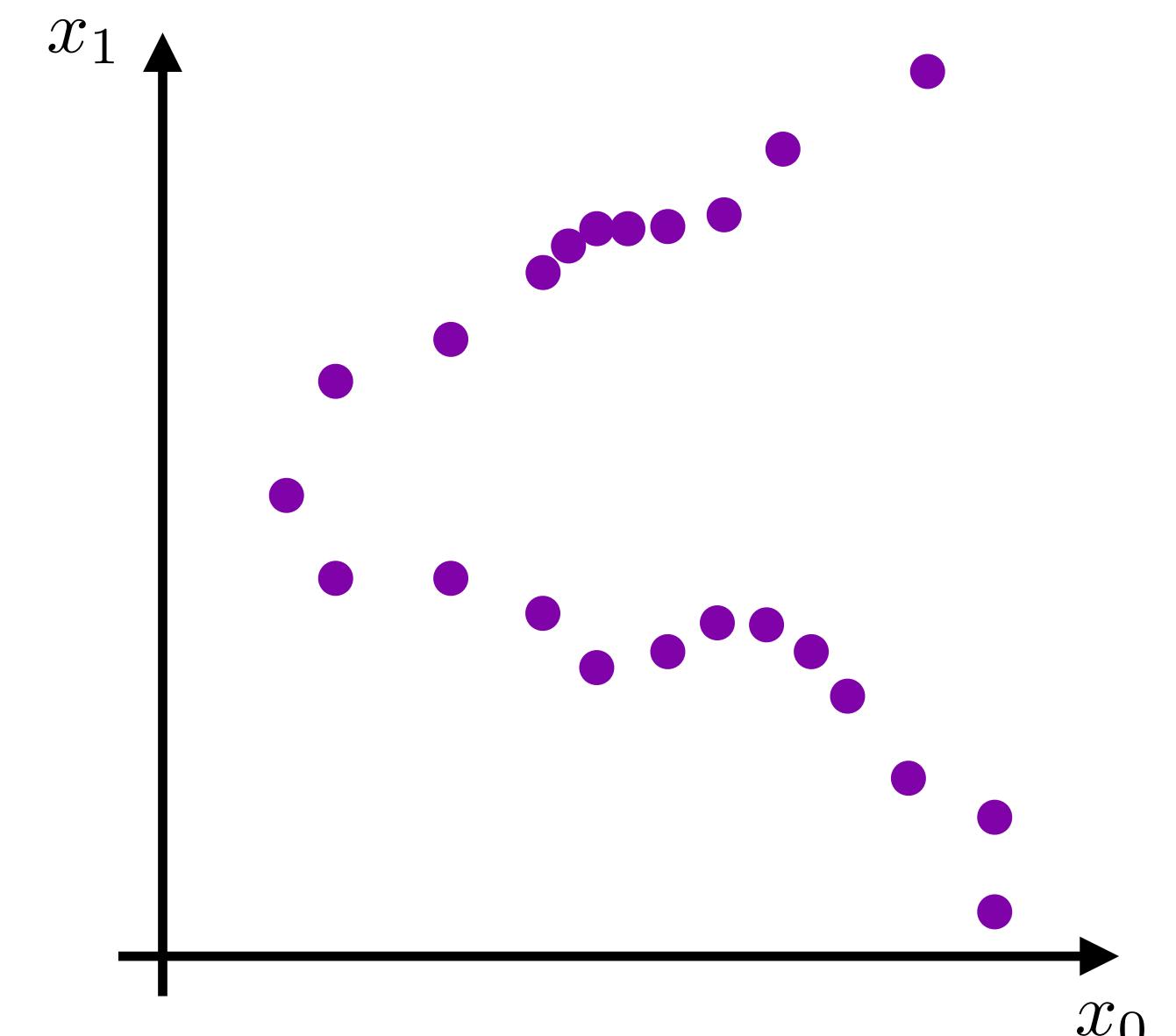


$$u \sim p_u(u)$$

n-dim. latent variables



unconstrained NN



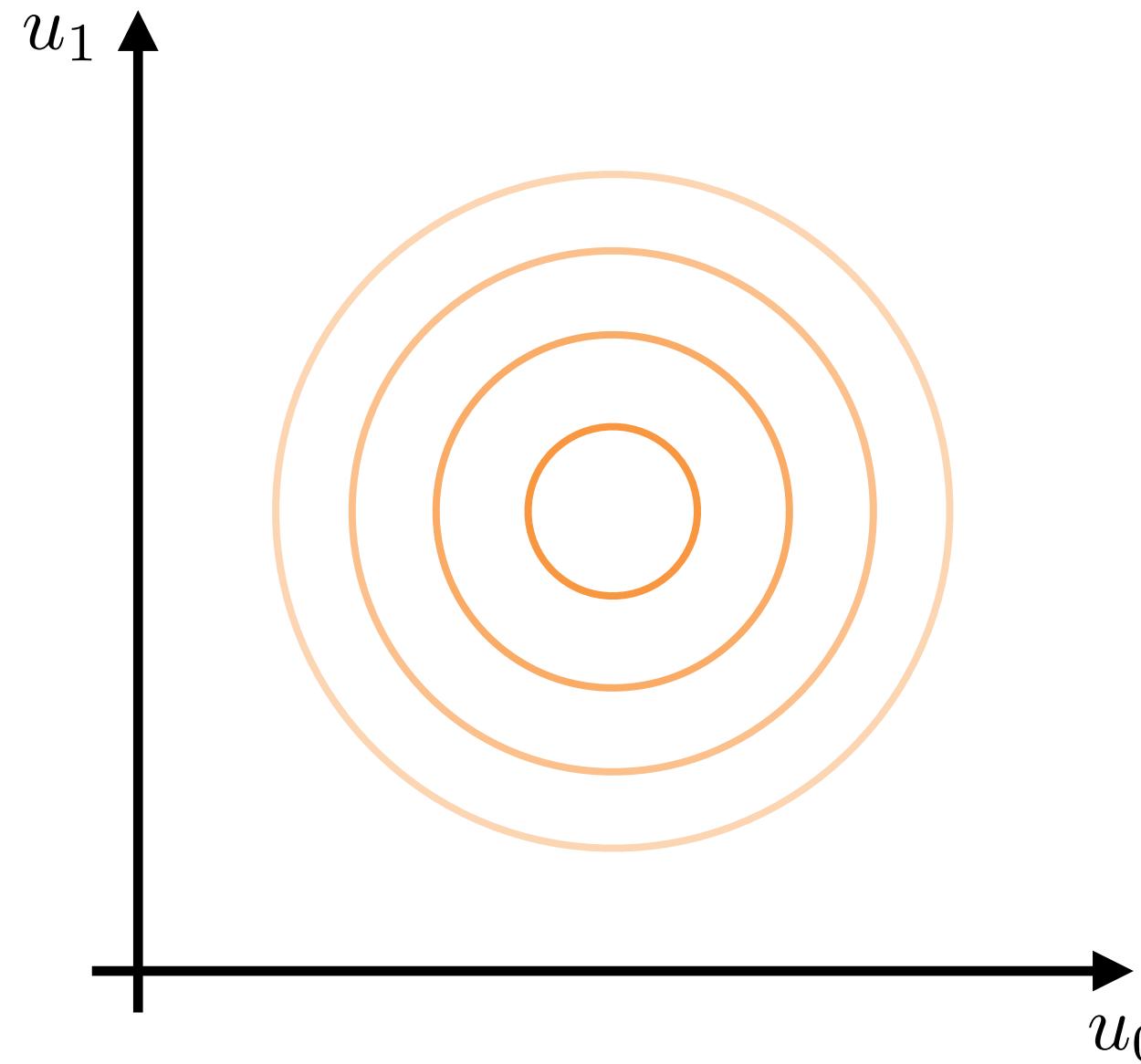
x

implicit density over \mathcal{M}

$p_{\mathcal{M}}(x)$ intractable

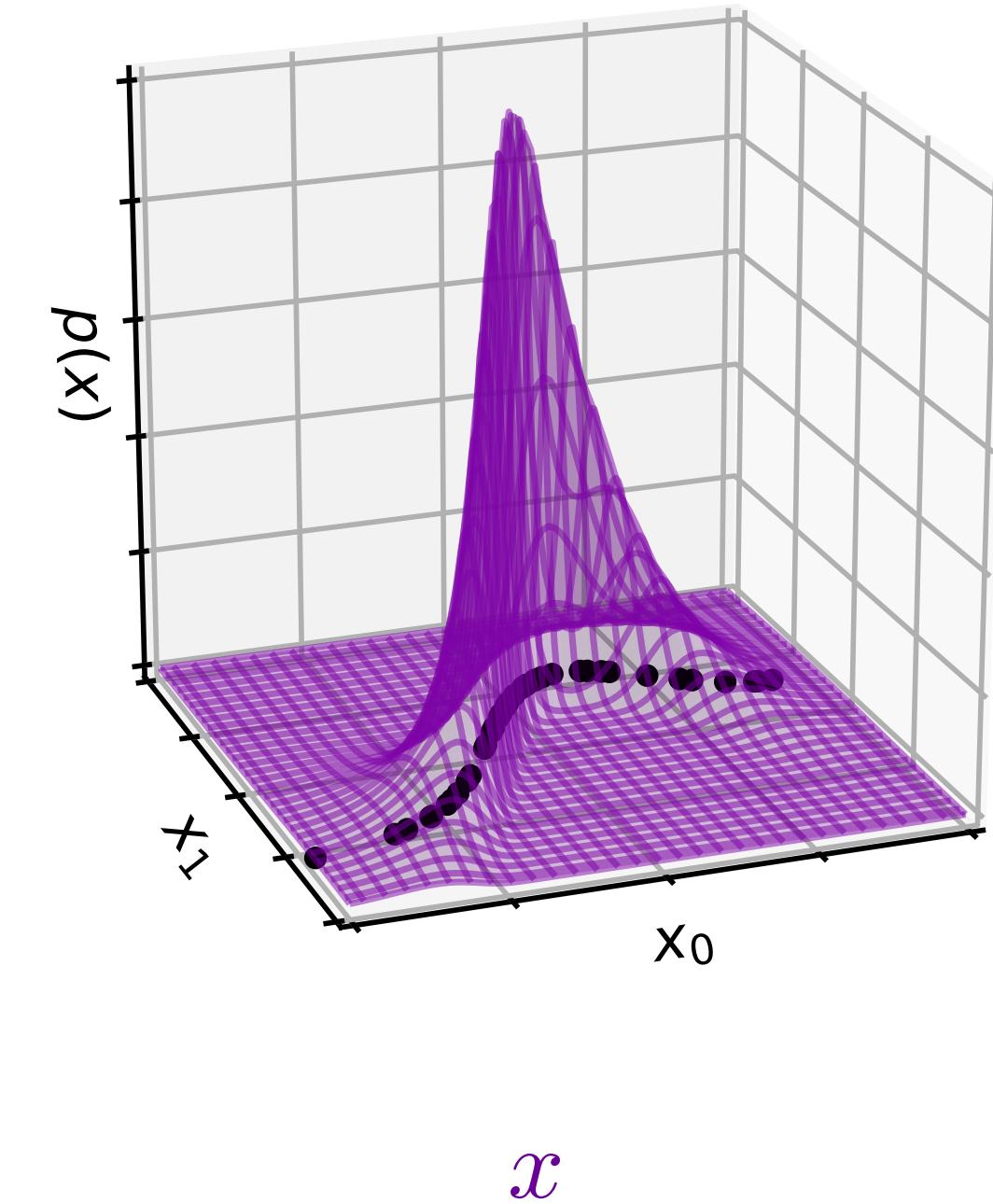
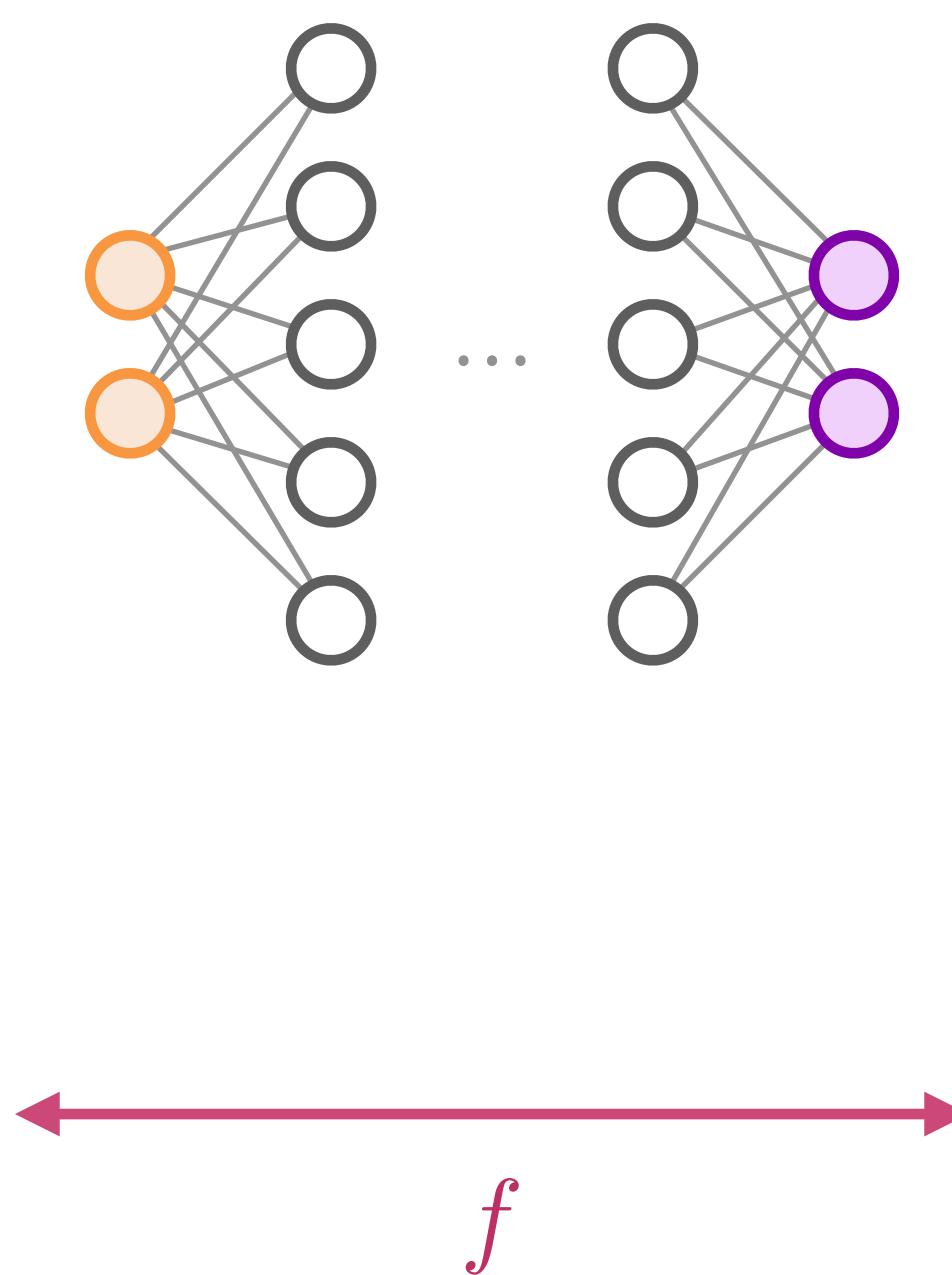
Normalizing flows in the ambient data space

[G. Papamakarios et al 1912.02762]



$$u \sim p_u(u)$$

d -dim. latent variables

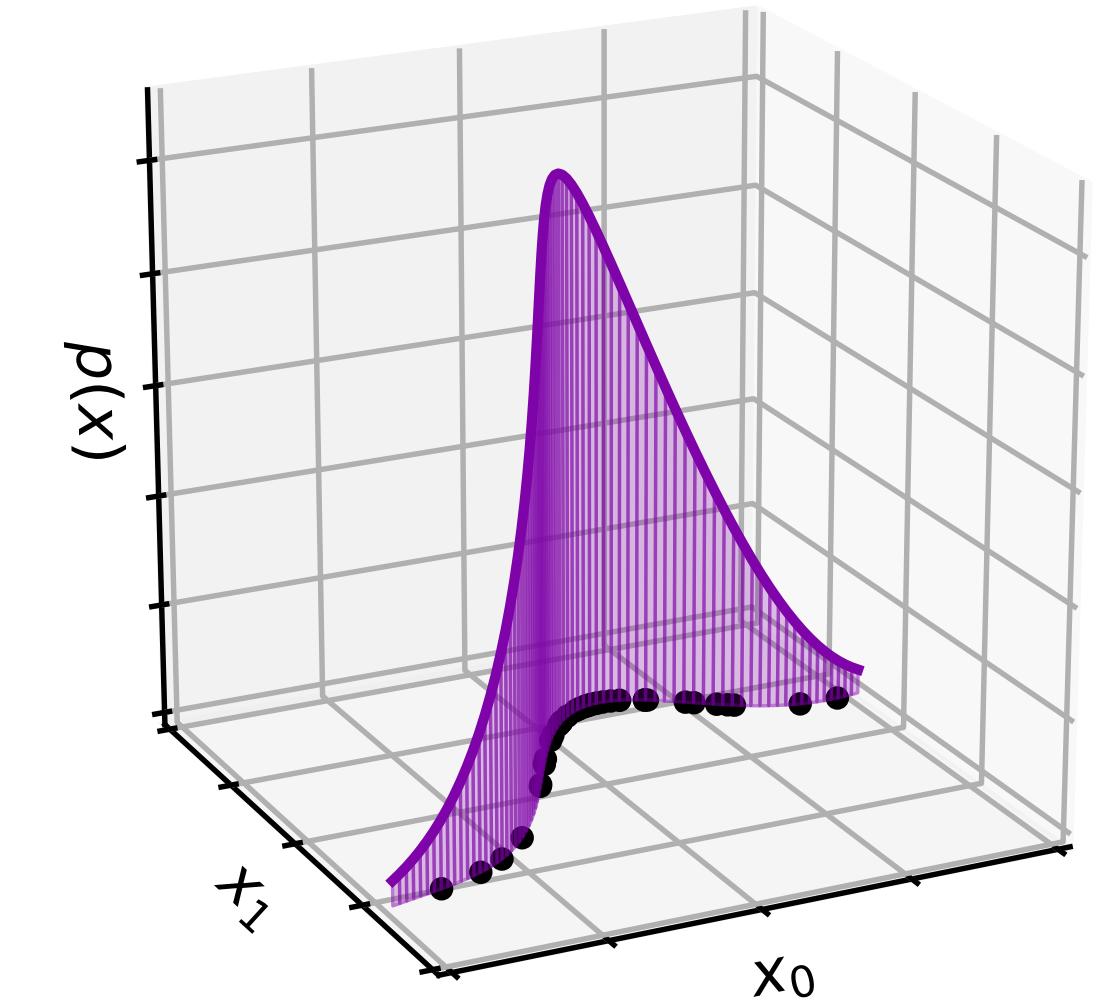
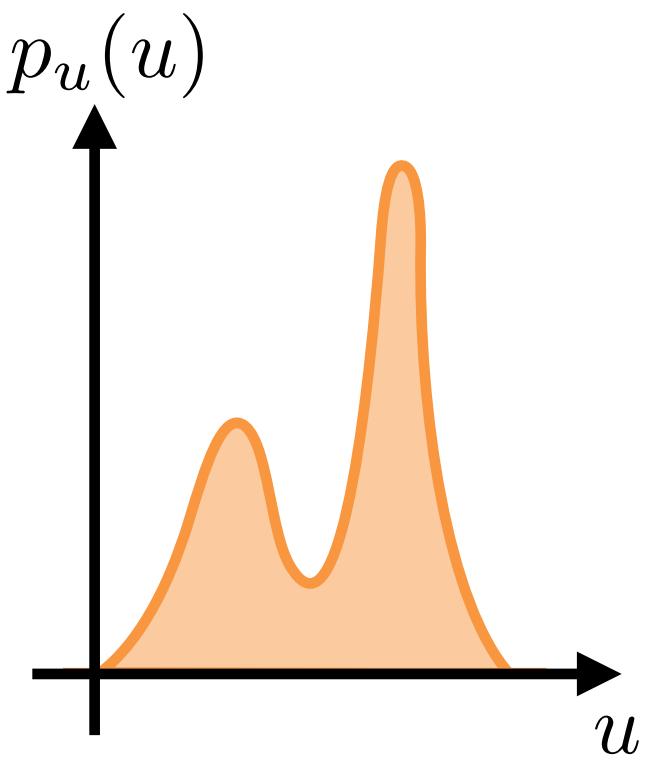
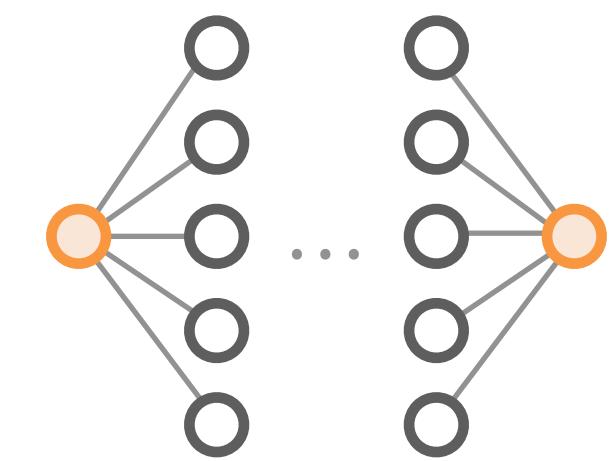
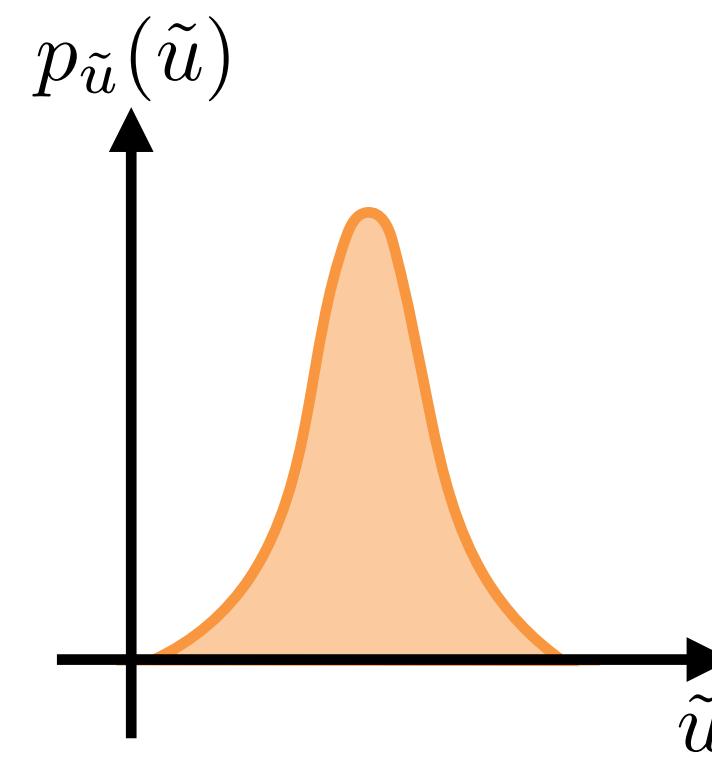


tractable density over
ambient data space

$$p_x(x) = p_u(f^{-1}(x)) |\det J_f(f^{-1}(x))|^{-1}$$

Flows on a prescribed manifold

[M. Gemici et al 1611.02304; D. Rezende et al 2002.02428]



$$\tilde{u} \sim p_{\tilde{u}}(\tilde{u})$$

$$\xleftarrow{h}$$

$$u$$

$$\xleftarrow{g^*}$$

n -dim. latents

invertible NN

n -dim. latents

prescribed chart

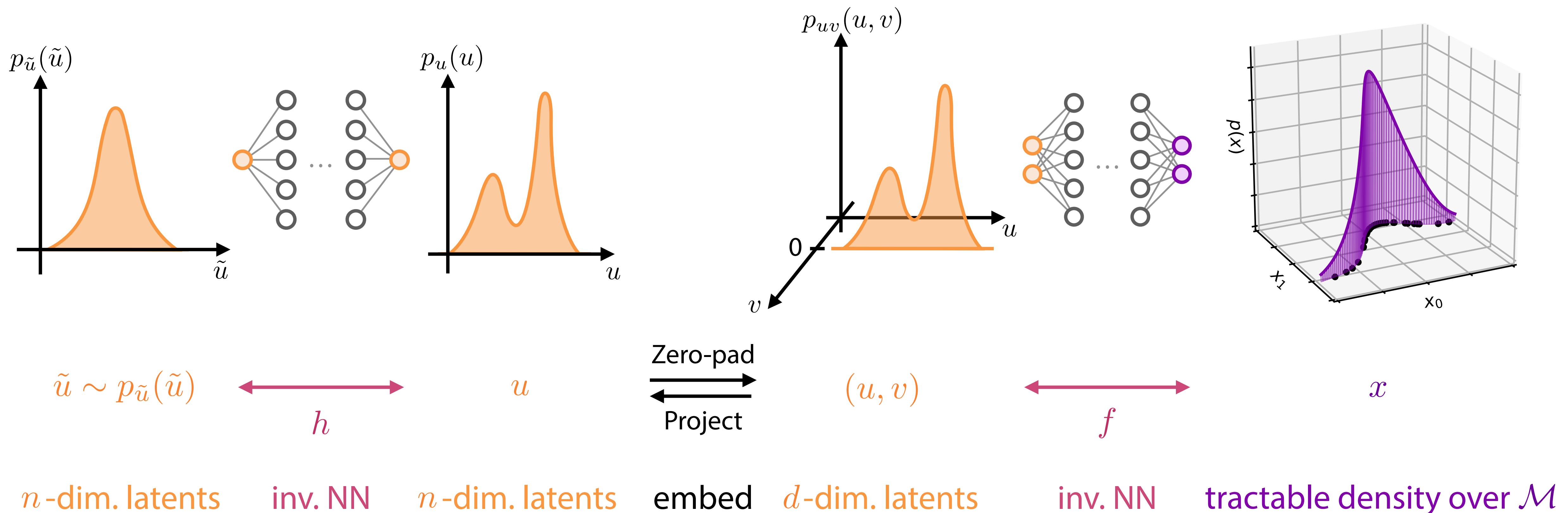
tractable density over \mathcal{M}^*

$$p_{\mathcal{M}^*}(x) = p_{\tilde{u}}(\tilde{u}) |\det J_h(\tilde{u})|^{-1}$$

$$\cdot |\det [J_{g^*}^T(u) J_{g^*}(u)]|^{-\frac{1}{2}}$$

\mathcal{M} -flows

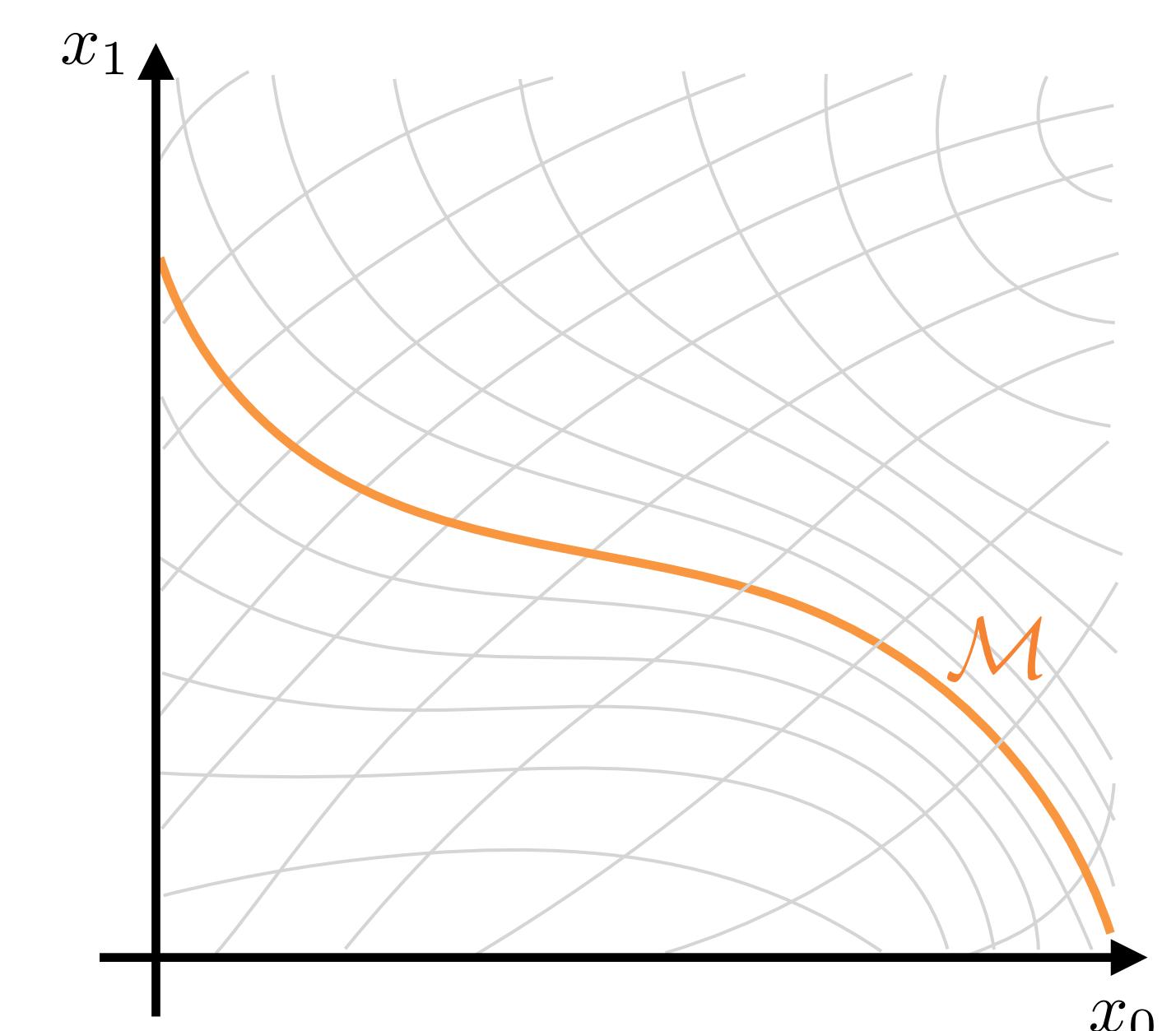
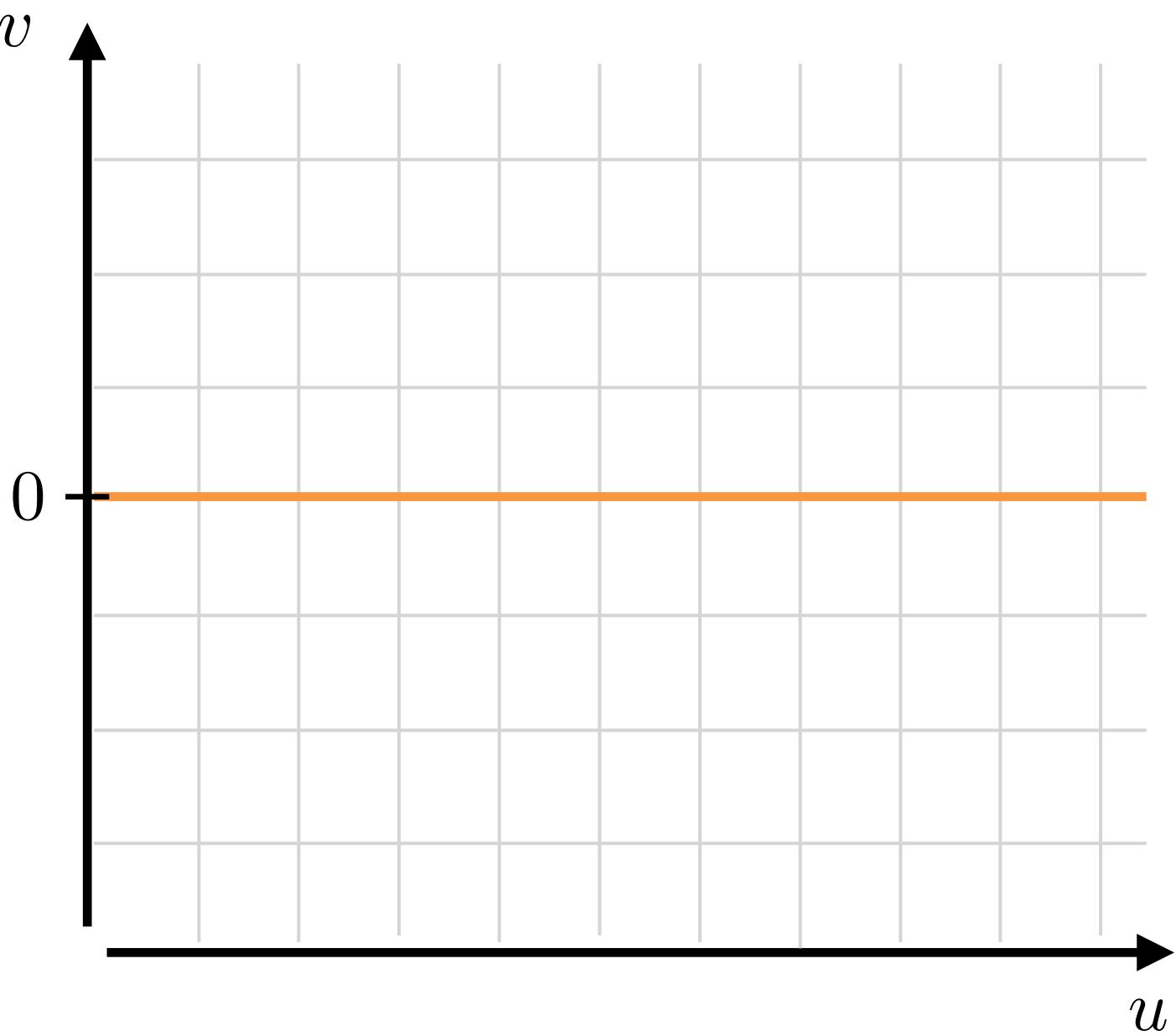
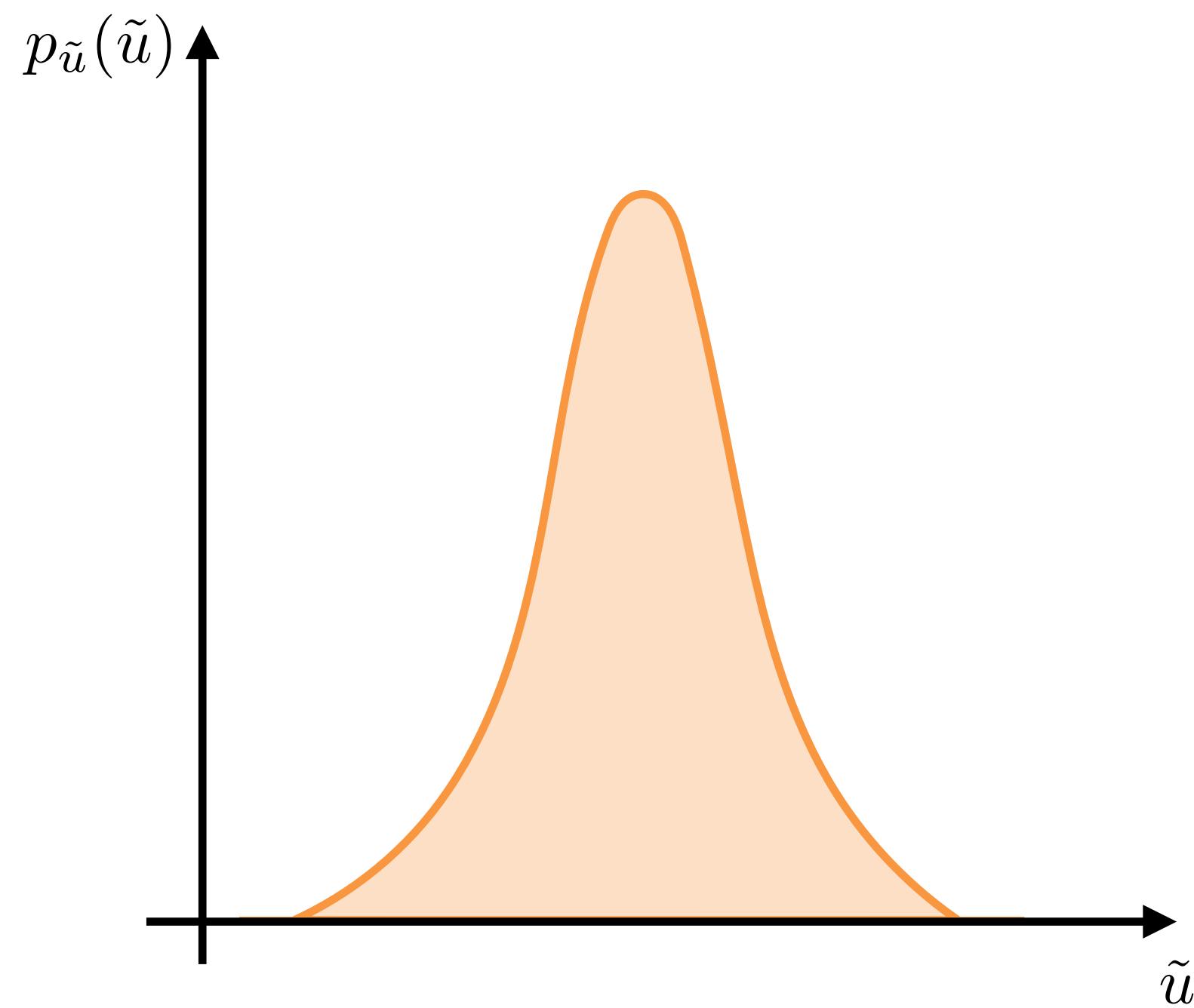
[JB, Kyle Cranmer 2003.13913]



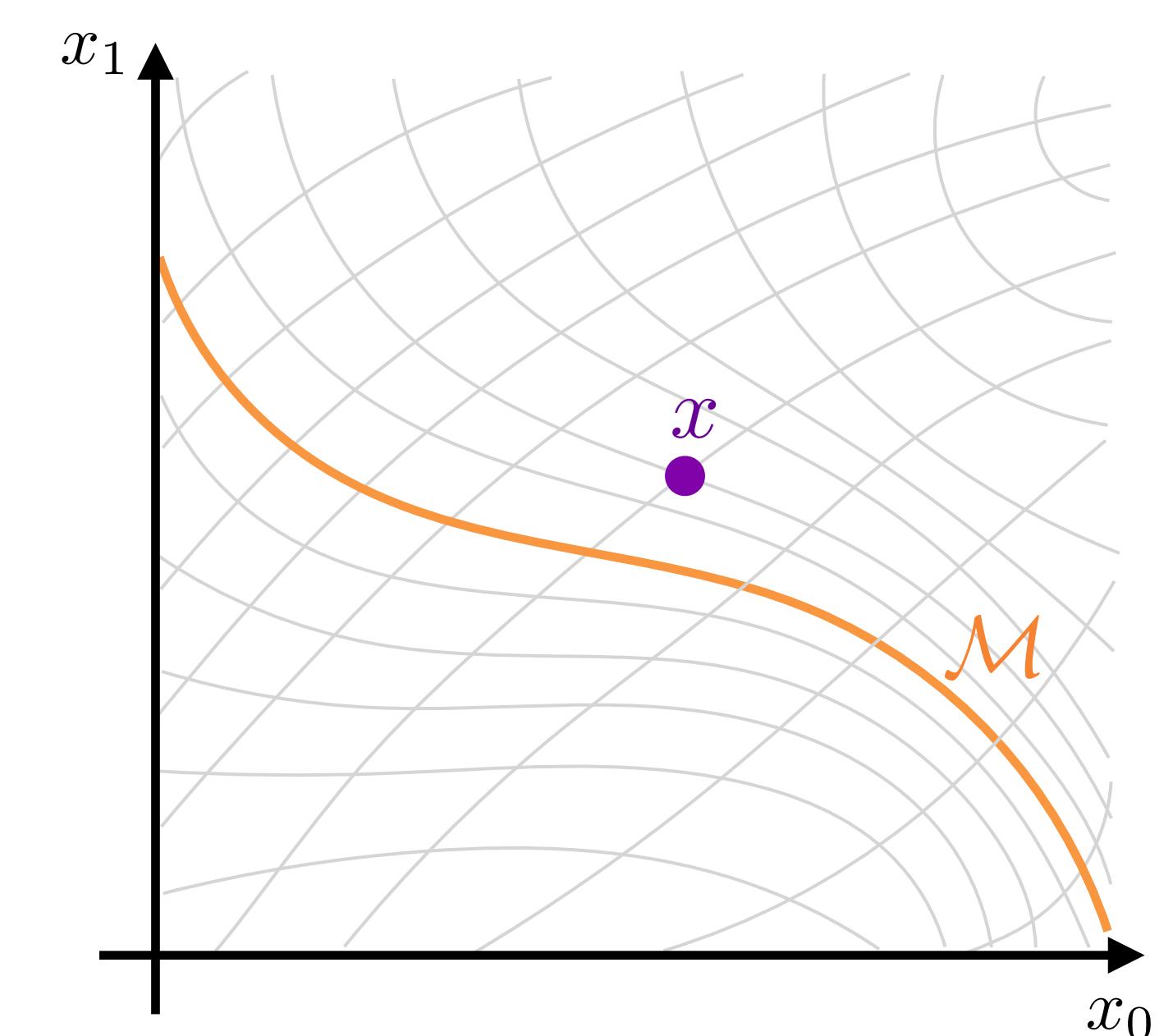
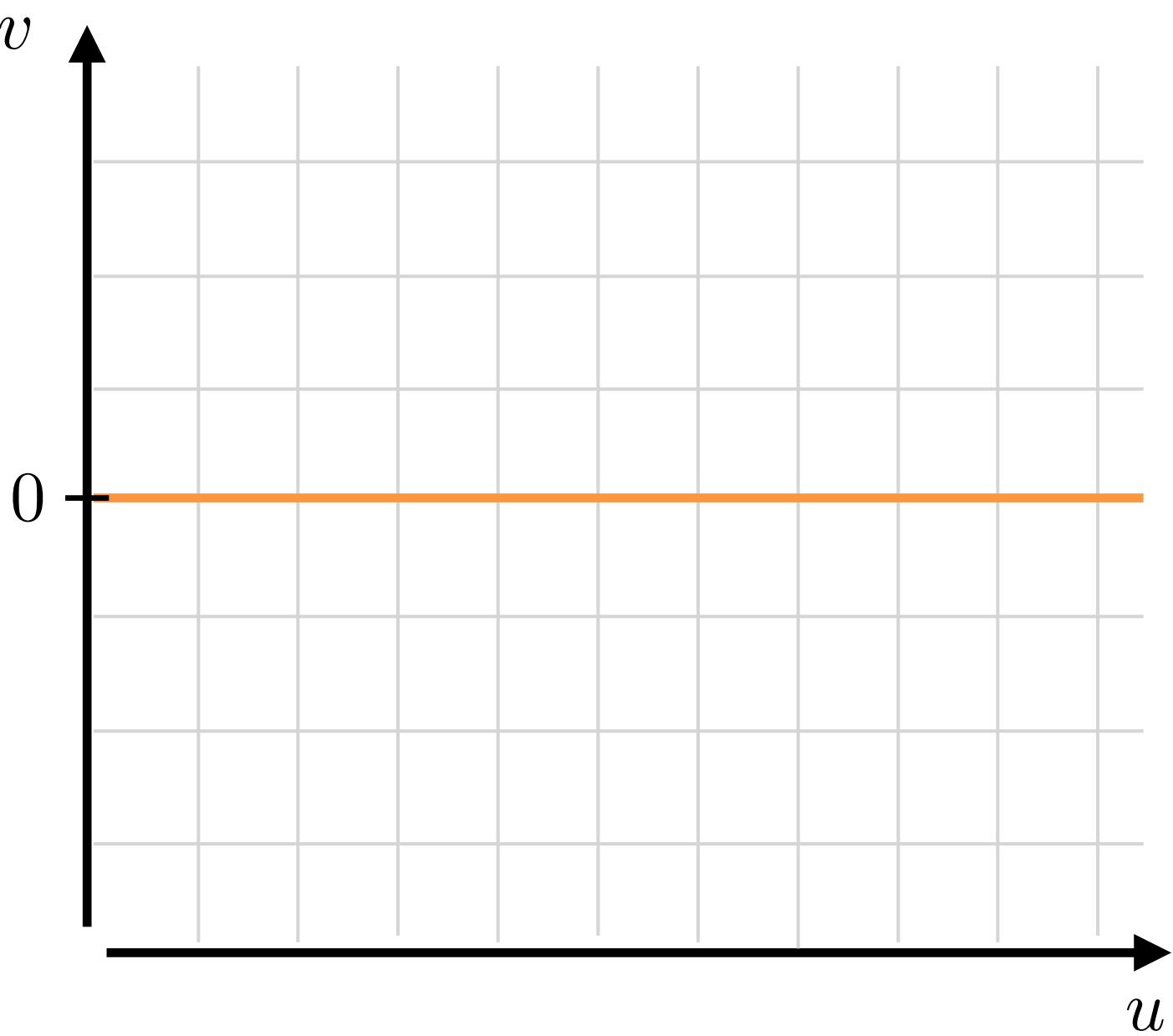
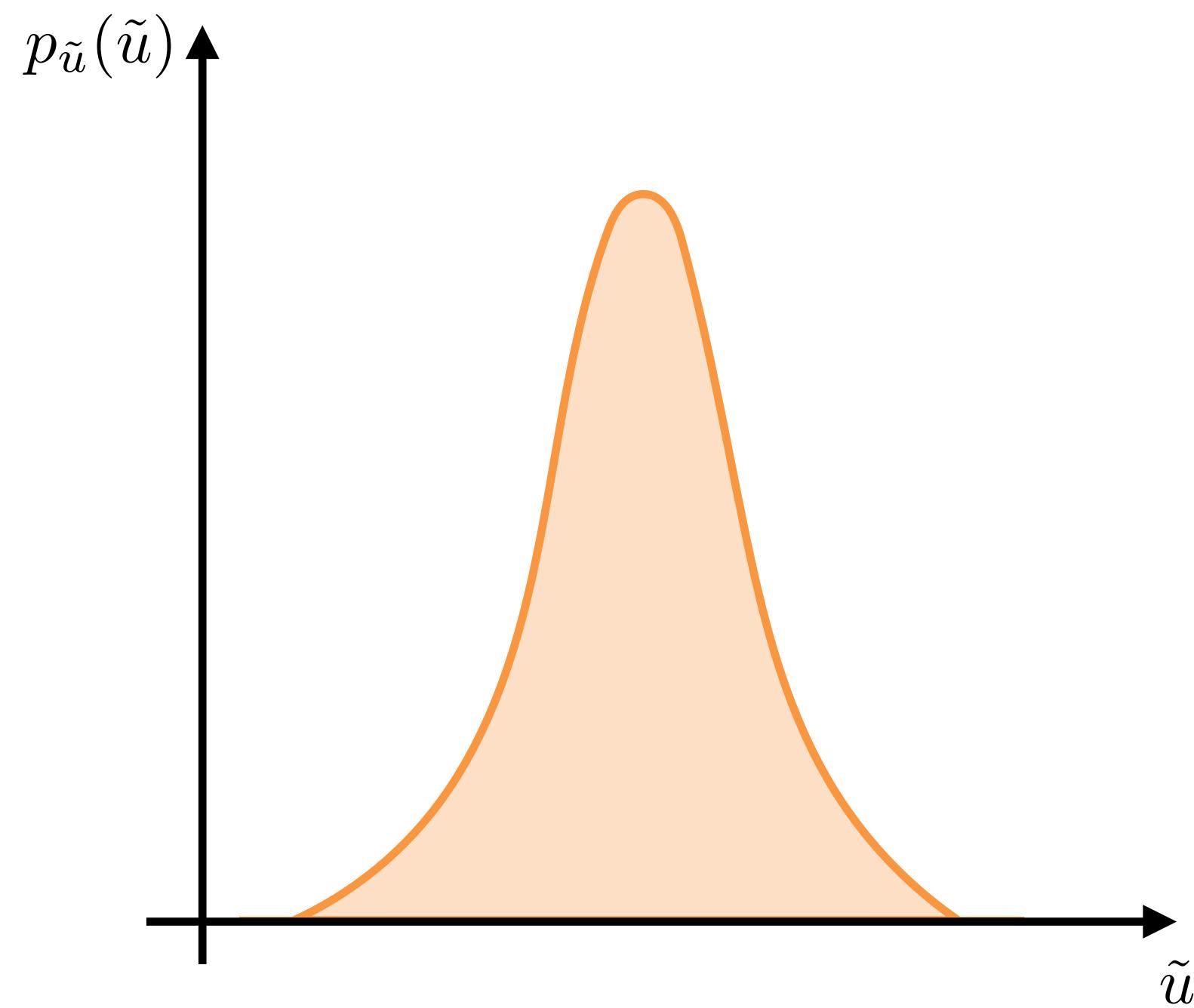
$$p_{\mathcal{M}}(x) = p_{\tilde{u}}(\tilde{u}) |\det J_h(\tilde{u})|^{-1}$$

$$\cdot \left| \det \left[(\mathbb{1} \ 0) J_f(u)^T J_f(u) \begin{pmatrix} \mathbb{1} \\ 0 \end{pmatrix} \right] \right|^{-\frac{1}{2}}$$

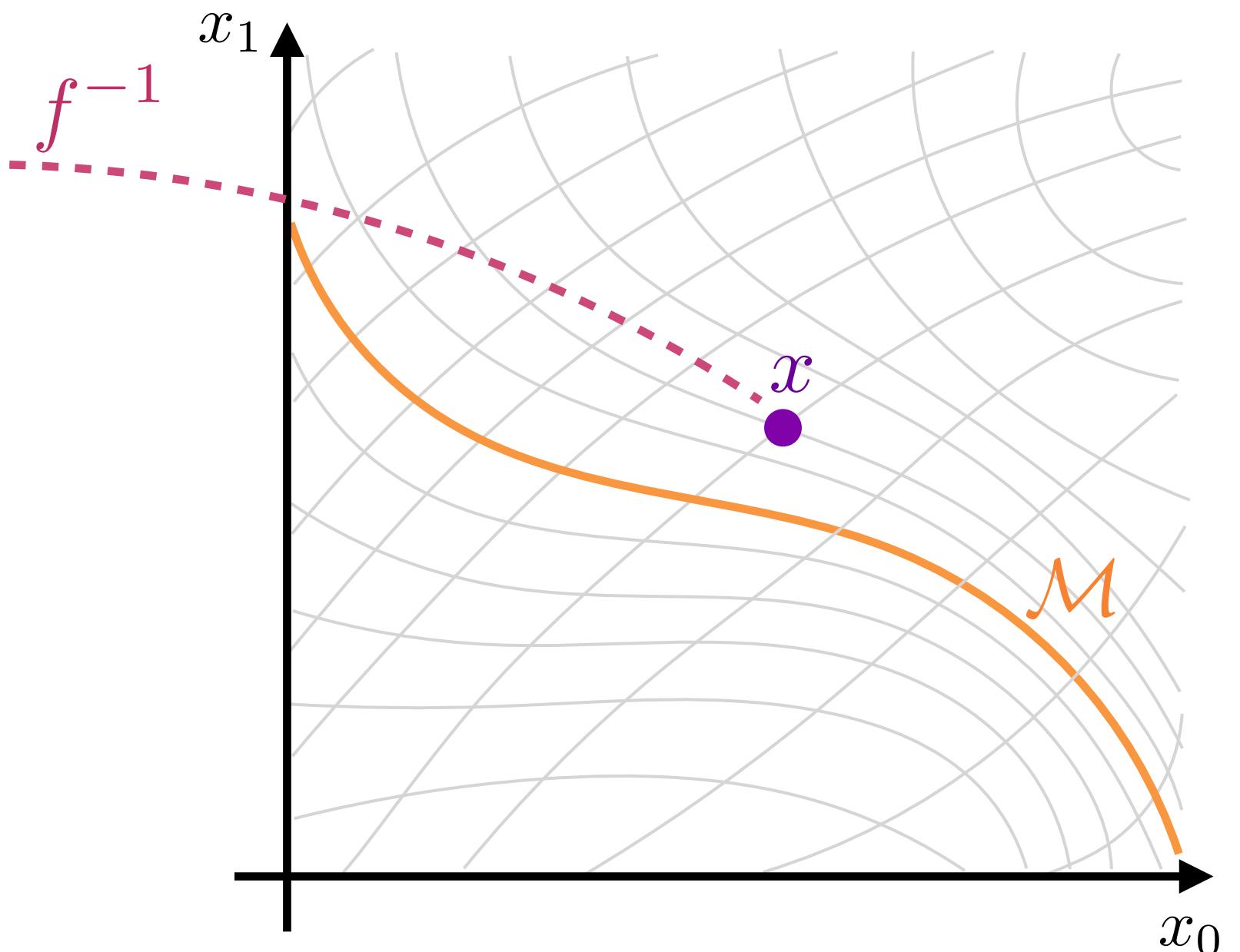
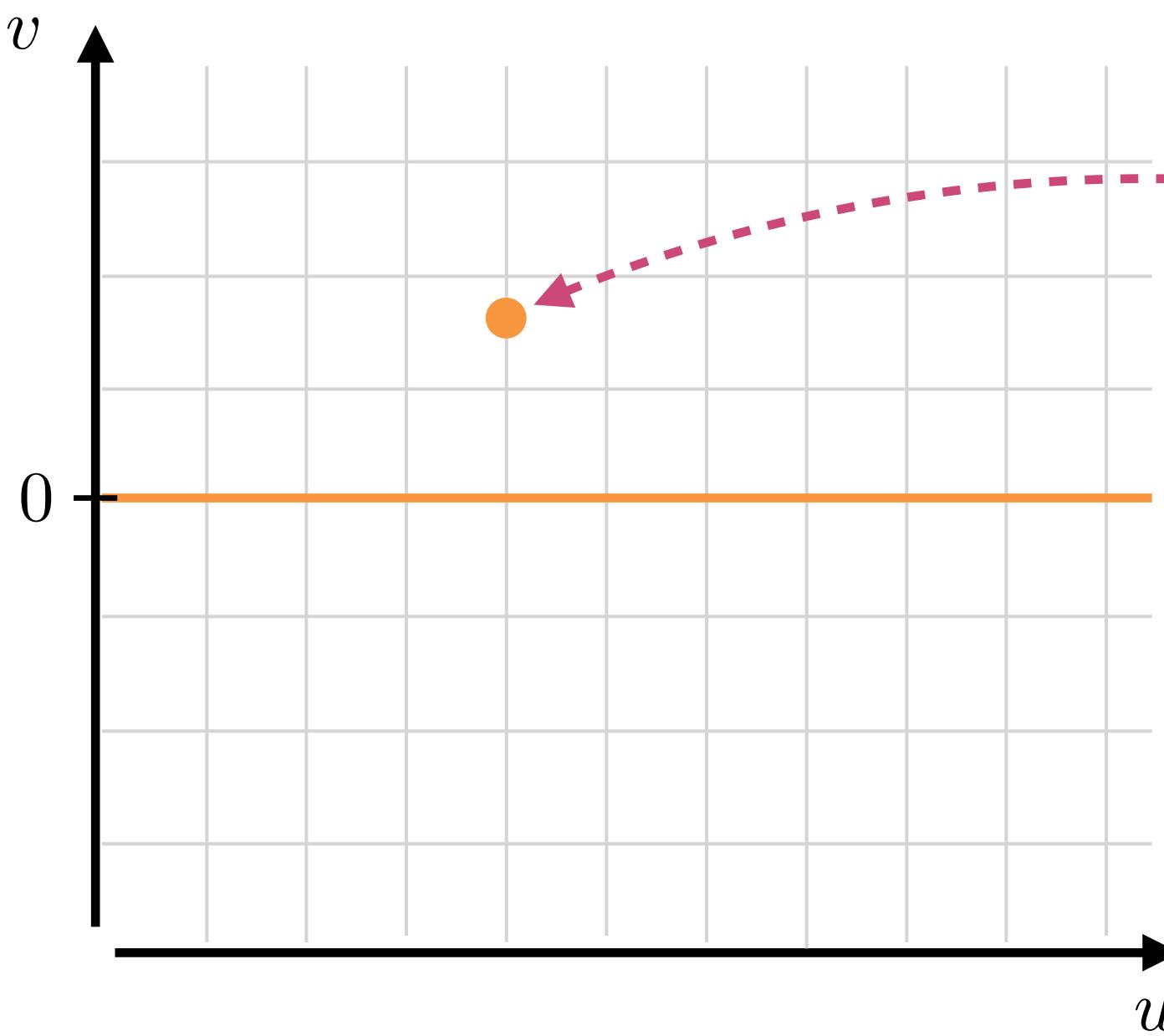
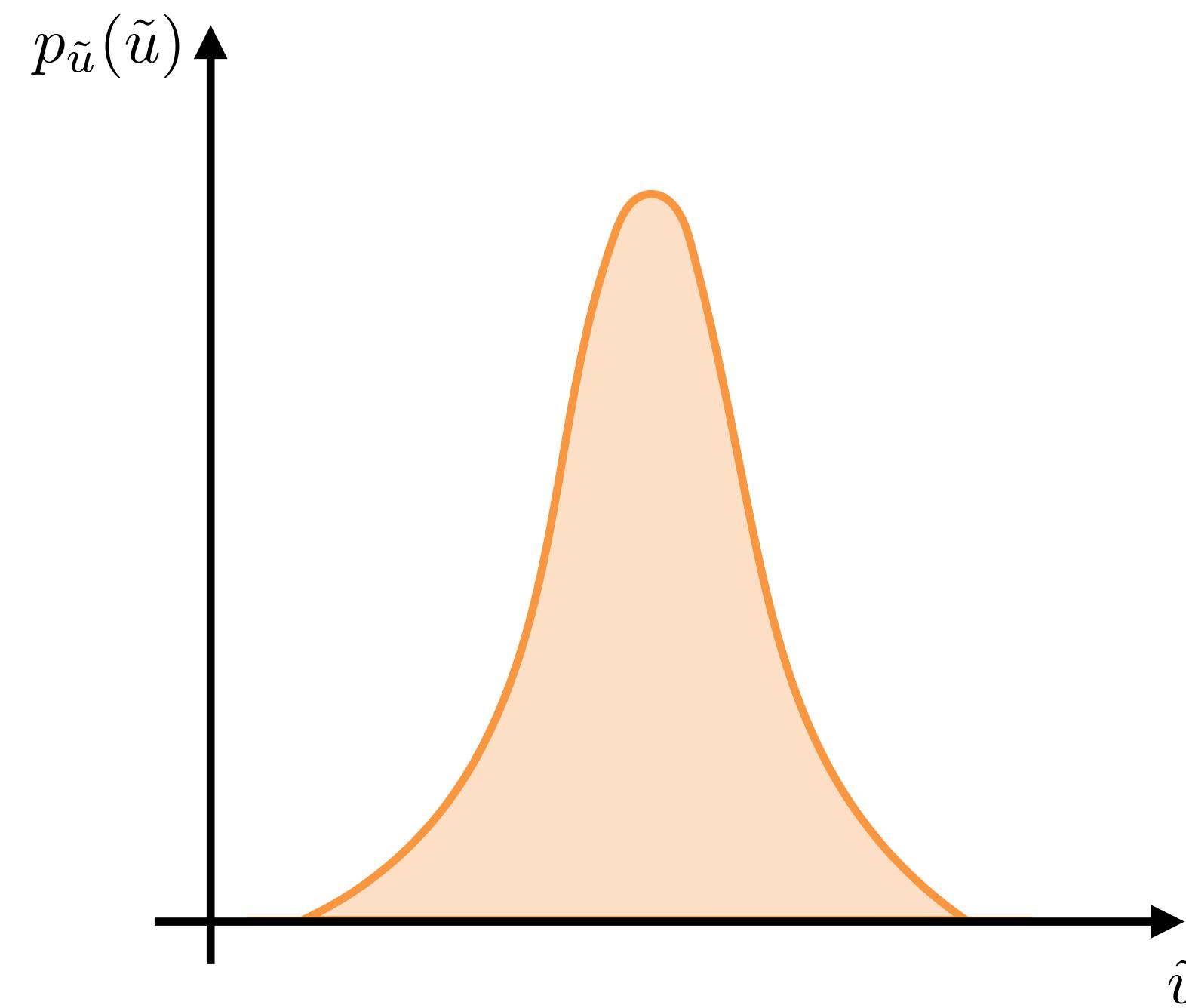
Evaluating data on or off the manifold



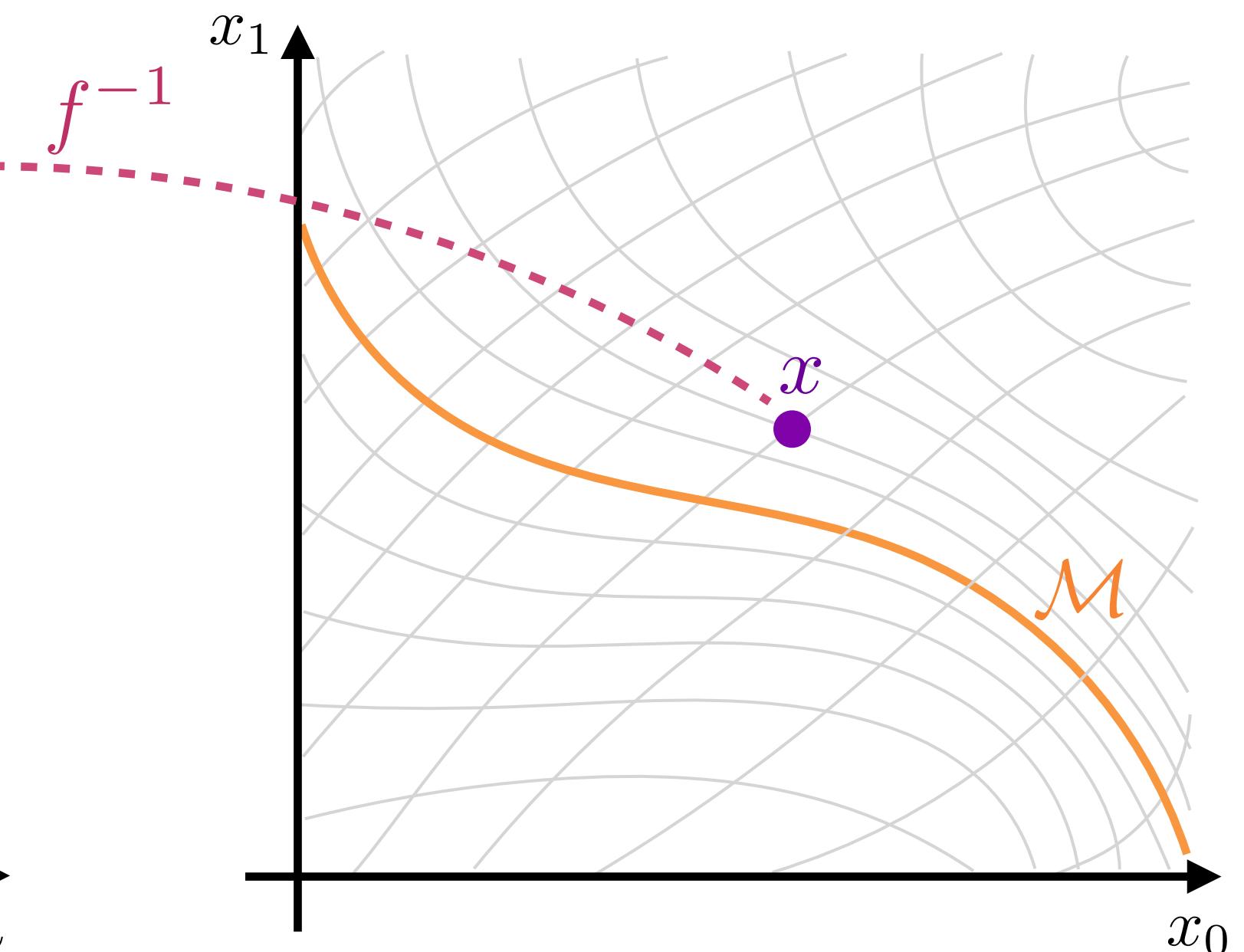
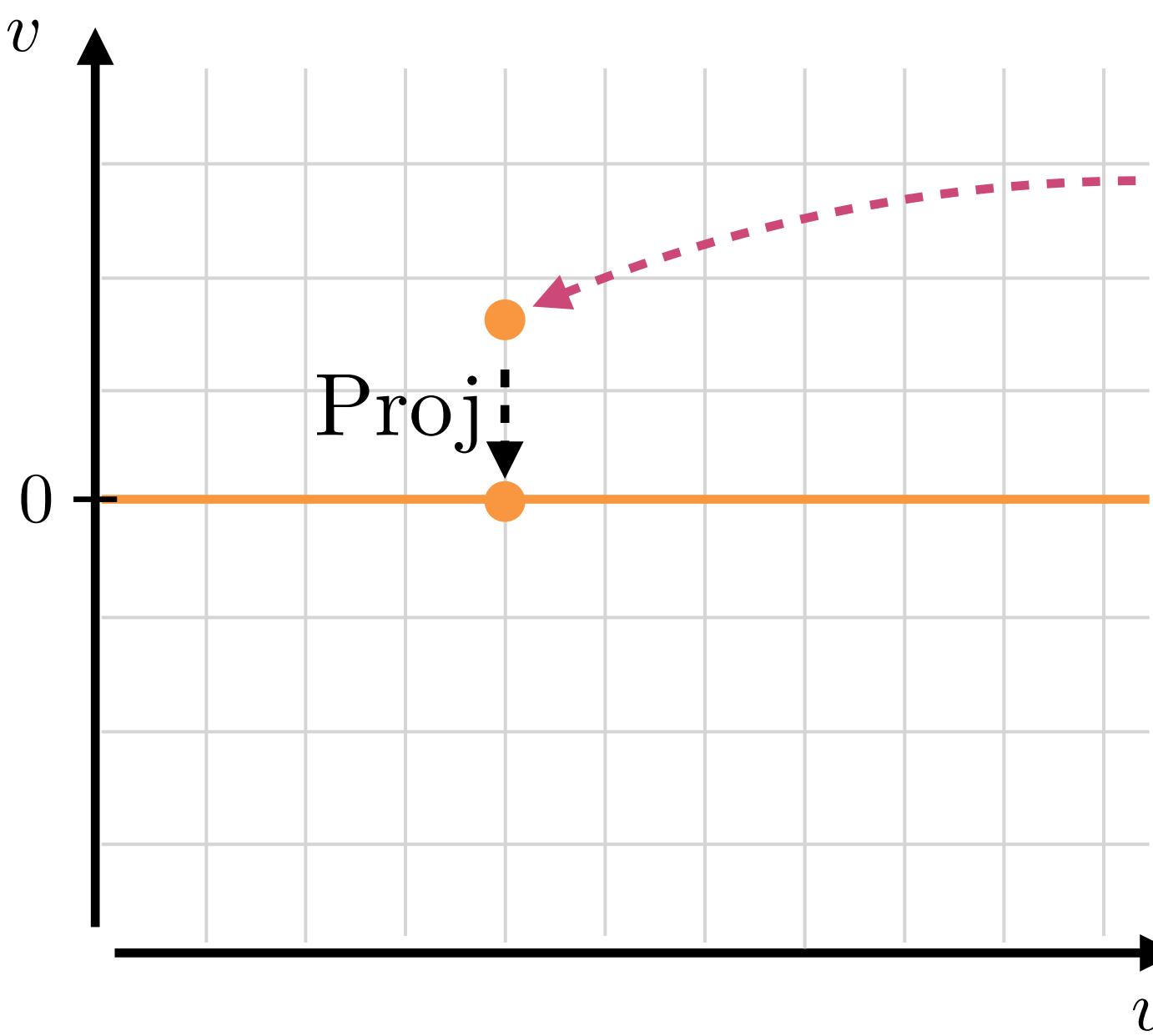
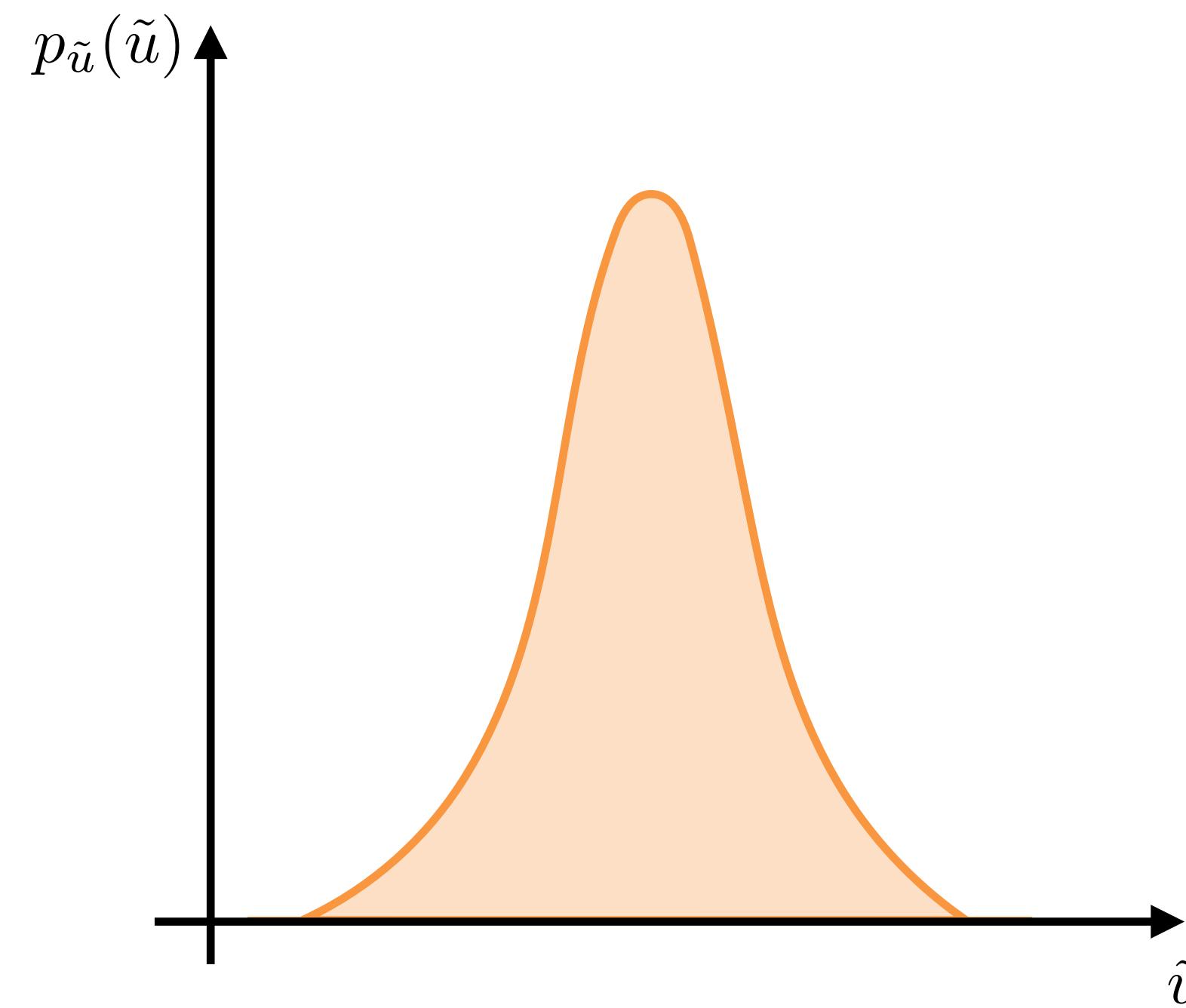
Evaluating data on or off the manifold



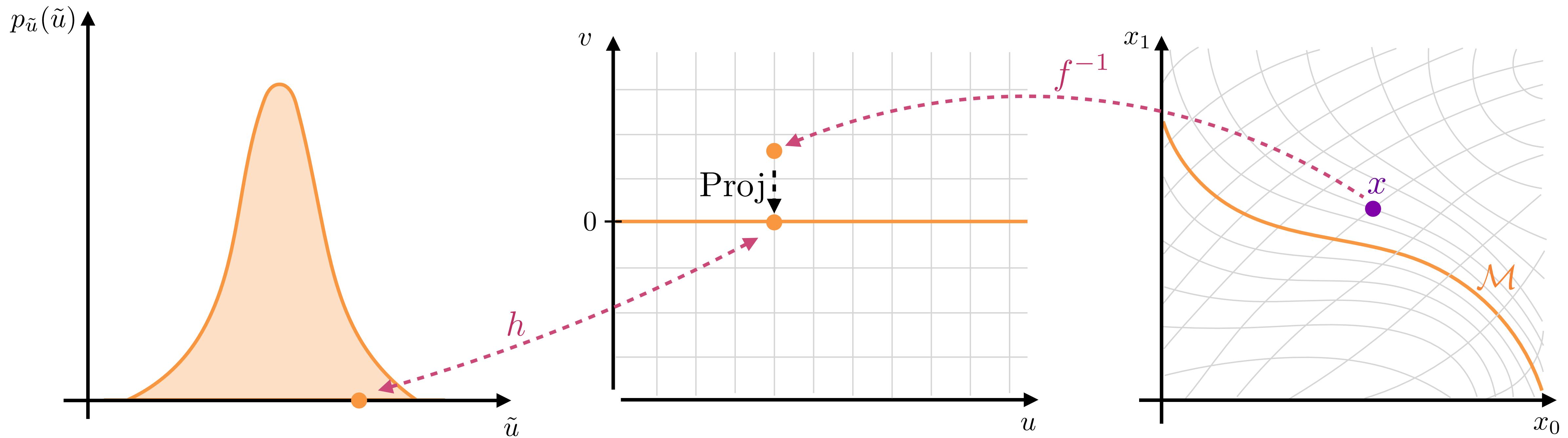
Evaluating data on or off the manifold



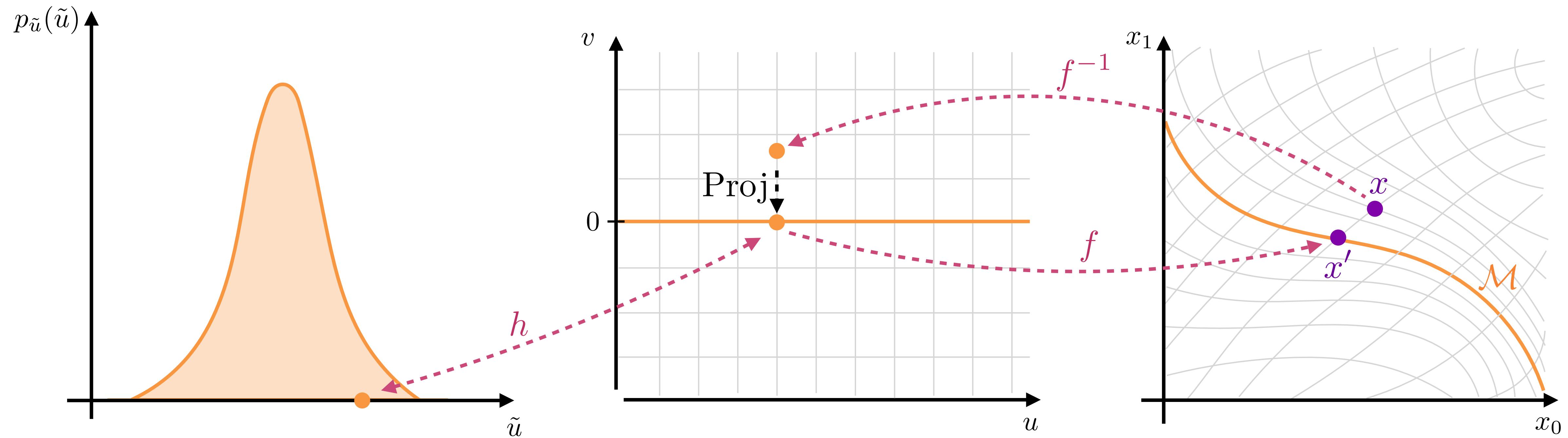
Evaluating data on or off the manifold



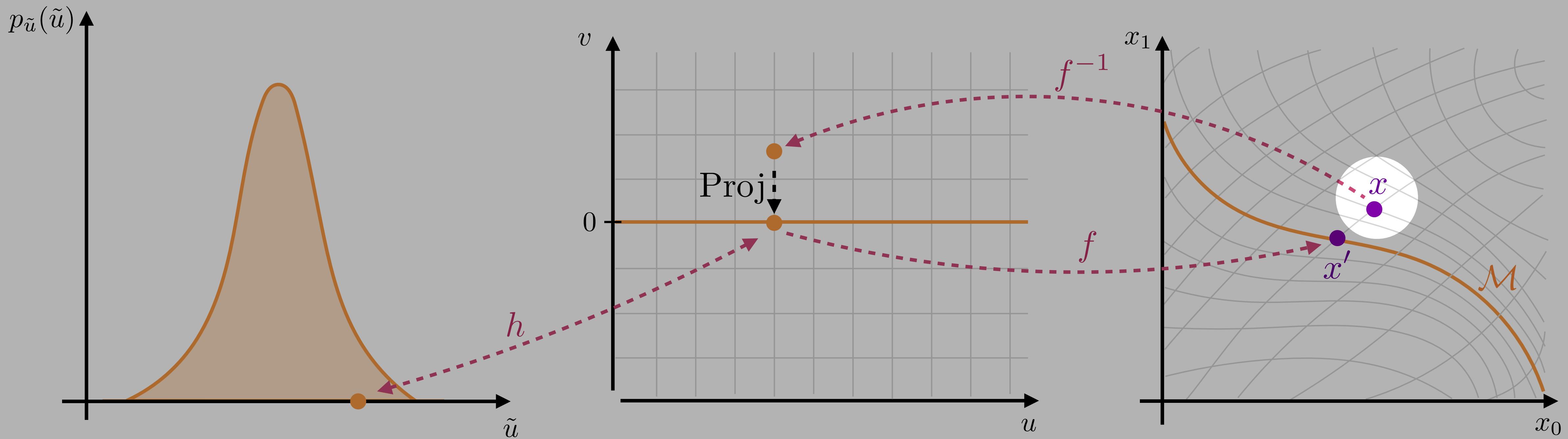
Evaluating data on or off the manifold



Evaluating data on or off the manifold

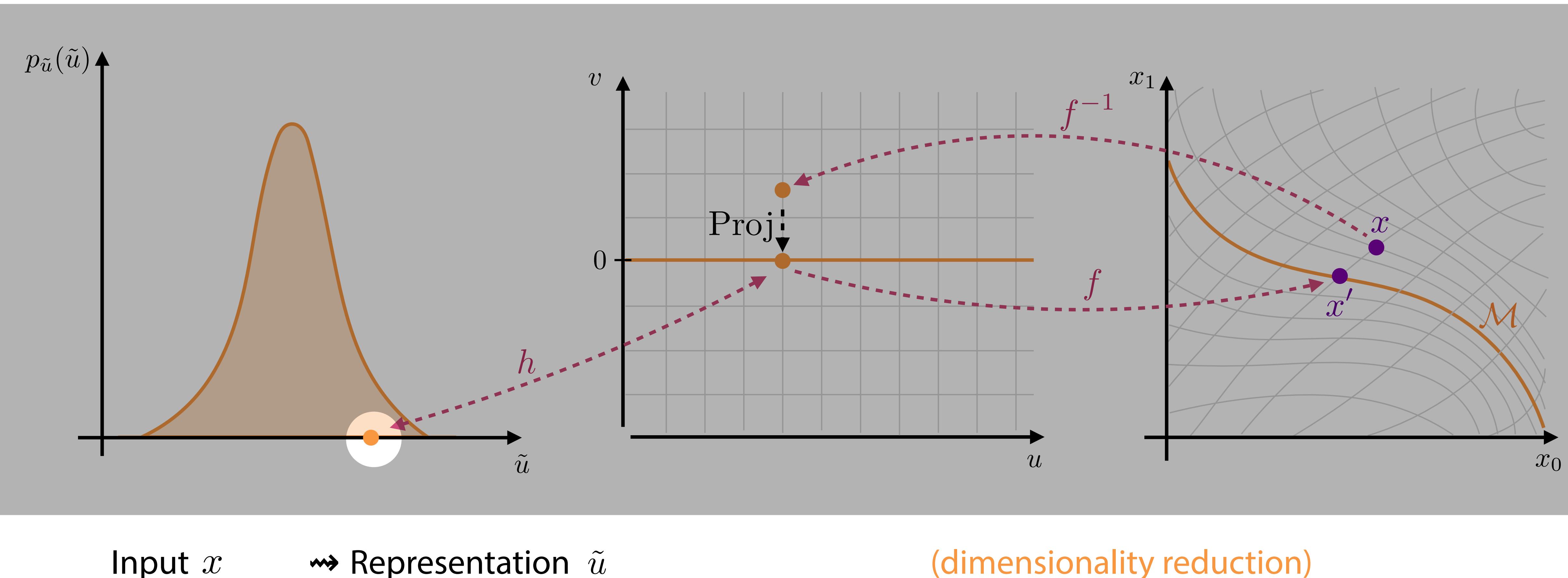


Evaluating data on or off the manifold

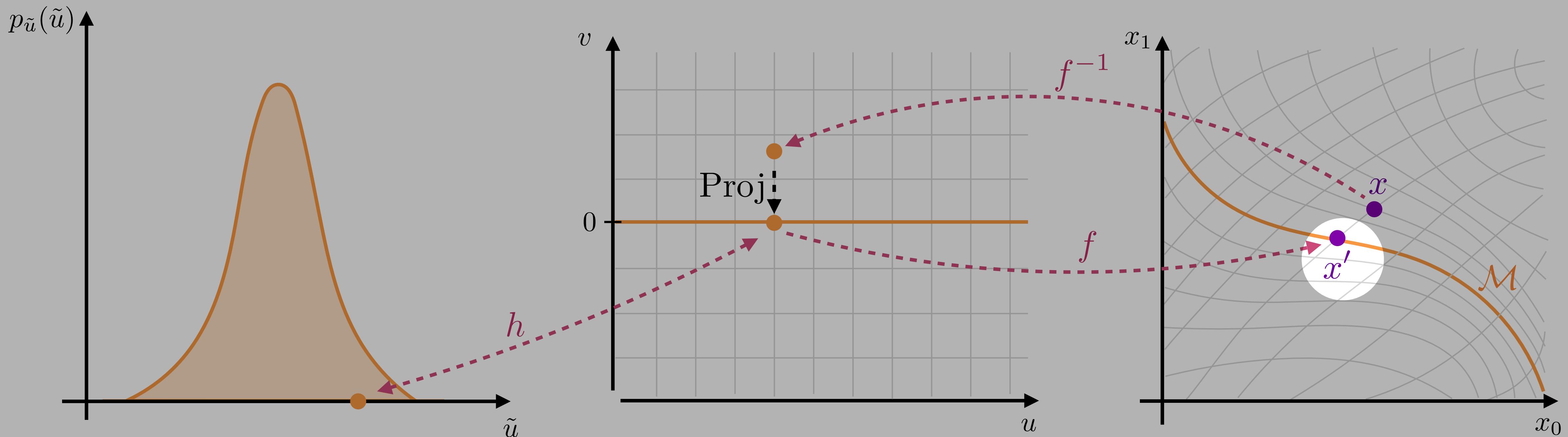


Input x

Evaluating data on or off the manifold



Evaluating data on or off the manifold



Input x

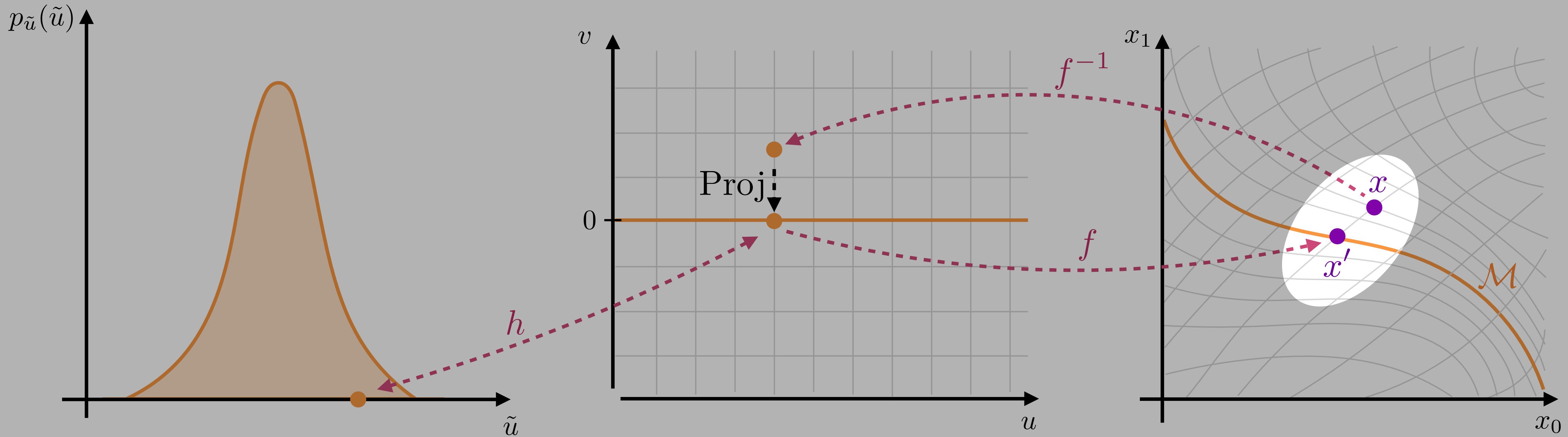
↔ Representation \tilde{u}

↔ Projection to manifold x'

(dimensionality reduction)

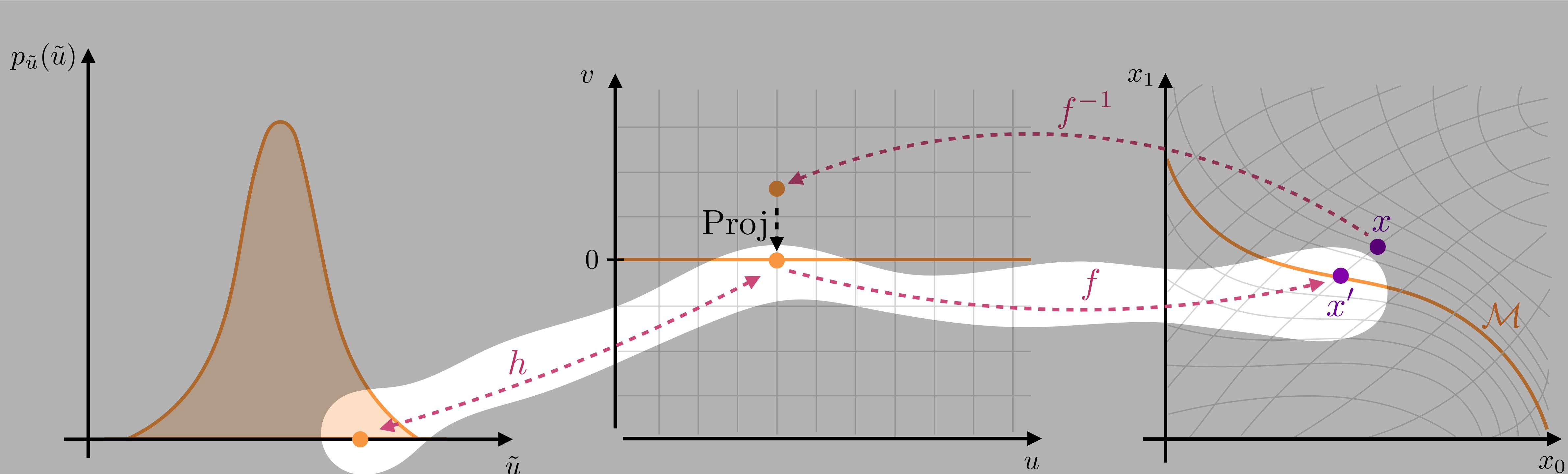
(denoising)

Evaluating data on or off the manifold



- | | | |
|-----------|-------------------------------------|----------------------------|
| Input x | ↔ Representation \tilde{u} | (dimensionality reduction) |
| | ↔ Projection to manifold x' | (denoising) |
| | ↔ Reconstruction error $\ x - x'\ $ | (training, OOD detection) |

Evaluating data on or off the manifold



- | | | |
|-----------|---|----------------------------|
| Input x | ↔ Representation \tilde{u} | (dimensionality reduction) |
| | ↔ Projection to manifold x' | (denoising) |
| | ↔ Reconstruction error $\ x - x'\ $ | (training, OOD detection) |
| | ↔ Likelihood after projection $p_{\mathcal{M}}(x')$ | (training, inference) |

Generative models vs. the data manifold

Model	Manifold	Chart	Generative	Tractable density	Restr. to manifold
Ambient flow (AF)	no	no	✓	✓	no
Flow on prescr. manifold	prescribed	prescribed	✓	✓	✓
GAN	learned	no	✓	no	✓
VAE	learned	no	✓	only ELBO	(no)
\mathcal{M} -flow	learned	learned	✓	✓ (potentially slow)	✓

Maximum likelihood is not enough

Likelihood defined after projection to \mathcal{M} ,
which is defined through NN weights ϕ_f

Family of likelihoods $p_{\phi_f}(x|\phi_h)$
rather than one likelihood $p(x|\phi_f, \phi_h)$

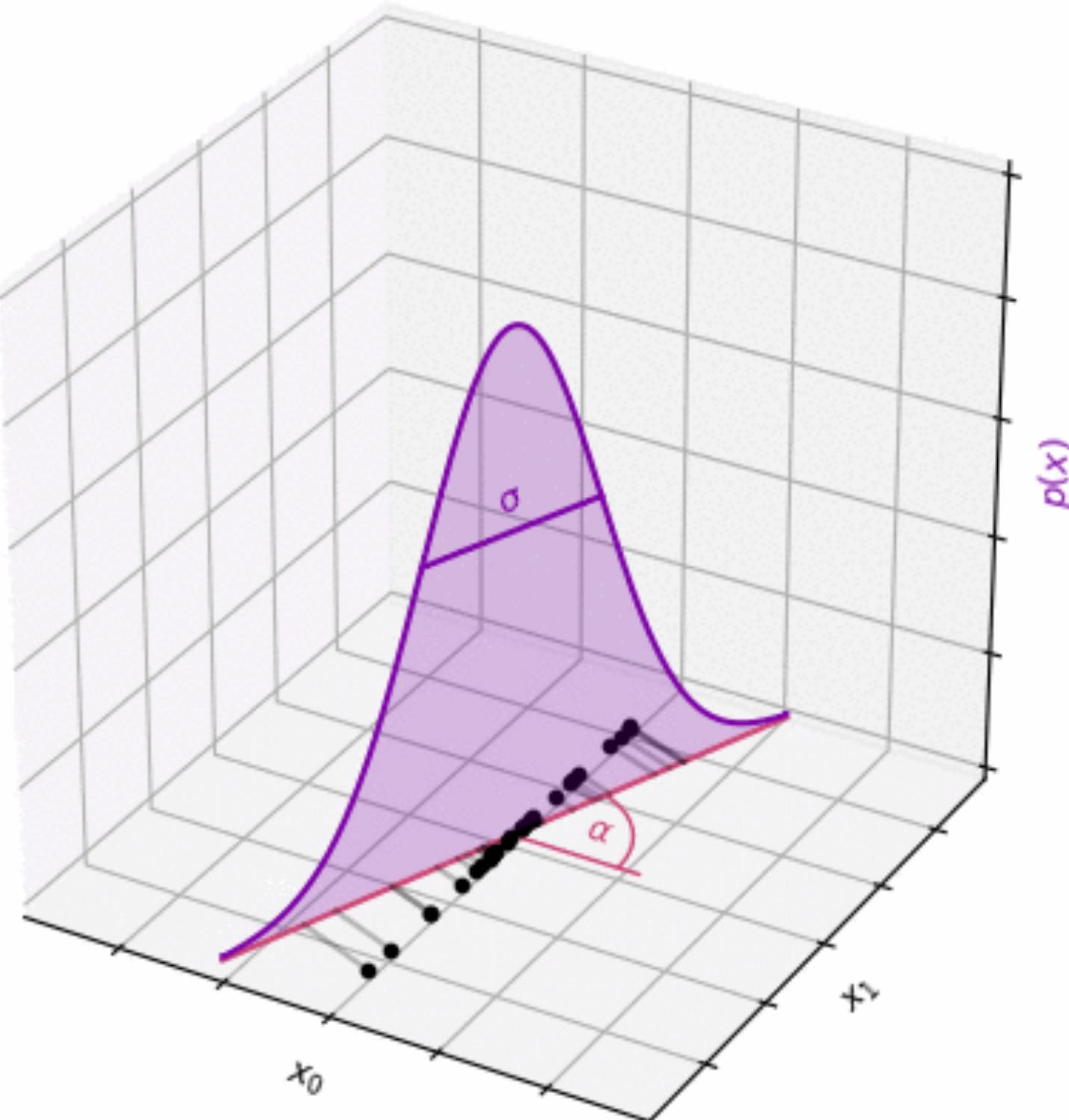
⇒ Learning ϕ_f by maximum
likelihood is unstable

Maximum likelihood is not enough

Likelihood defined after projection to \mathcal{M} ,
which is defined through NN weights ϕ_f

Family of likelihoods $p_{\phi_f}(x|\phi_h)$
rather than one likelihood $p(x|\phi_f, \phi_h)$

⇒ Learning ϕ_f by maximum
likelihood is unstable

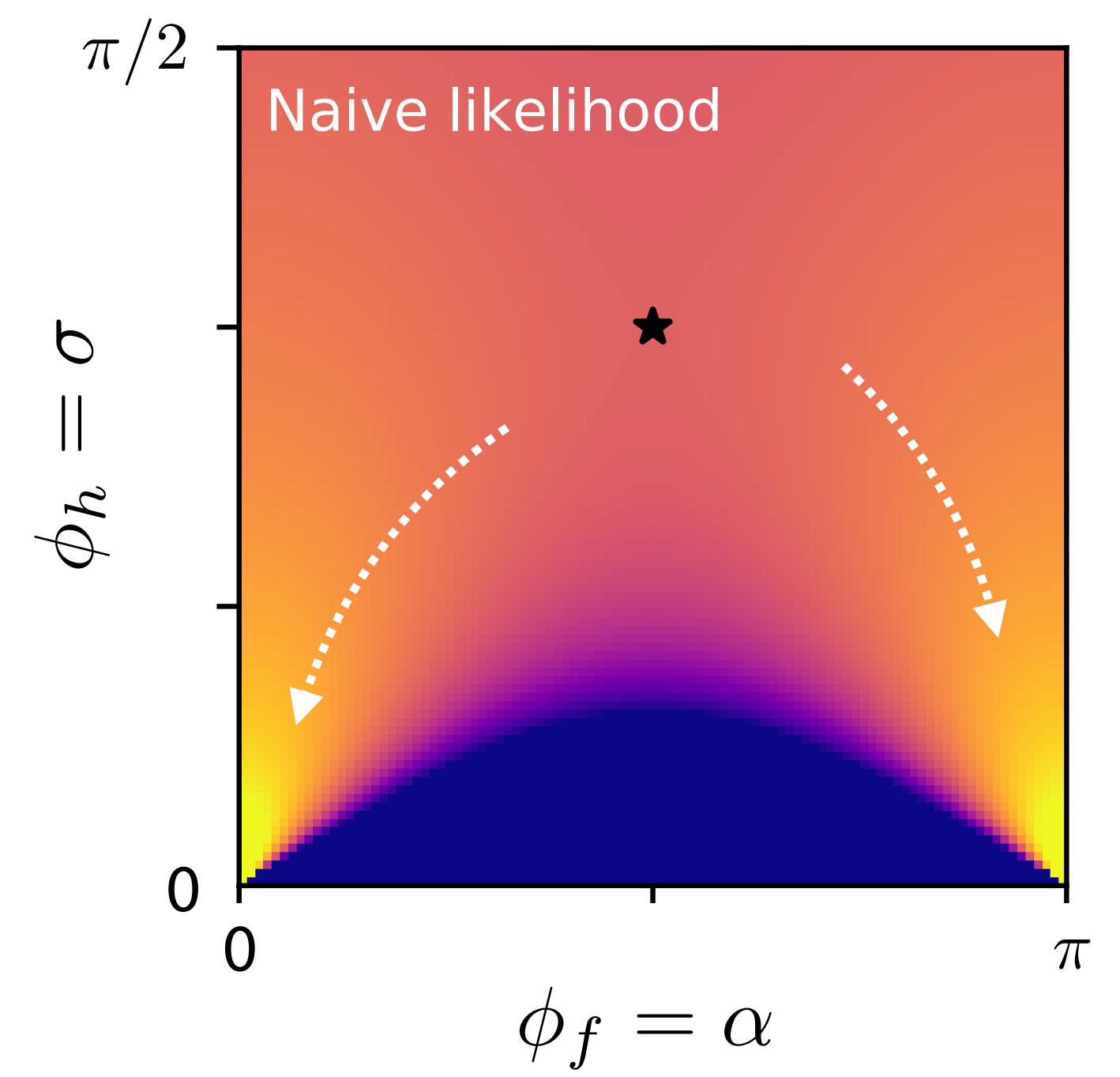
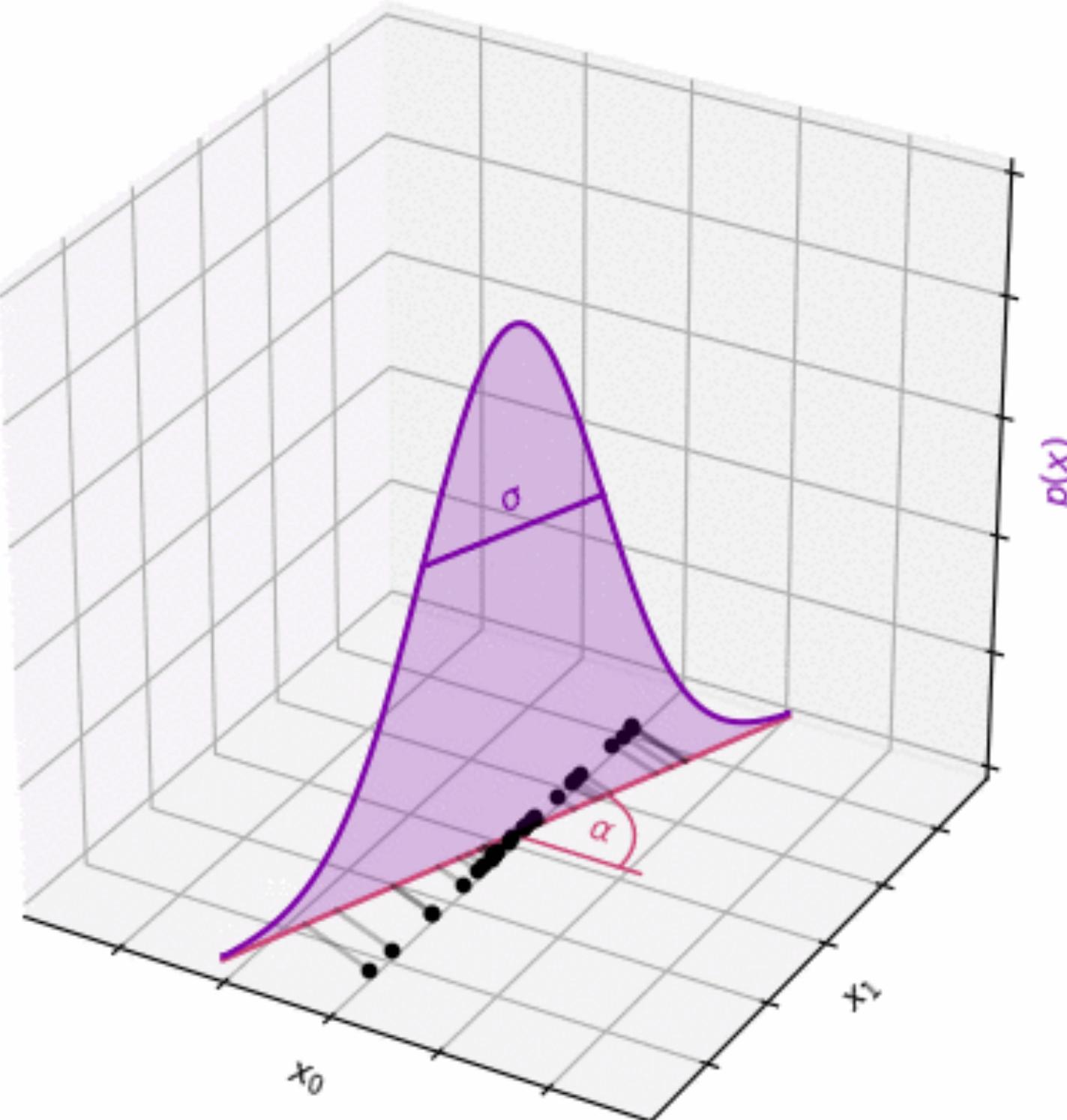


Maximum likelihood is not enough

Likelihood defined after projection to \mathcal{M} ,
which is defined through NN weights ϕ_f

Family of likelihoods $p_{\phi_f}(x|\phi_h)$
rather than one likelihood $p(x|\phi_f, \phi_h)$

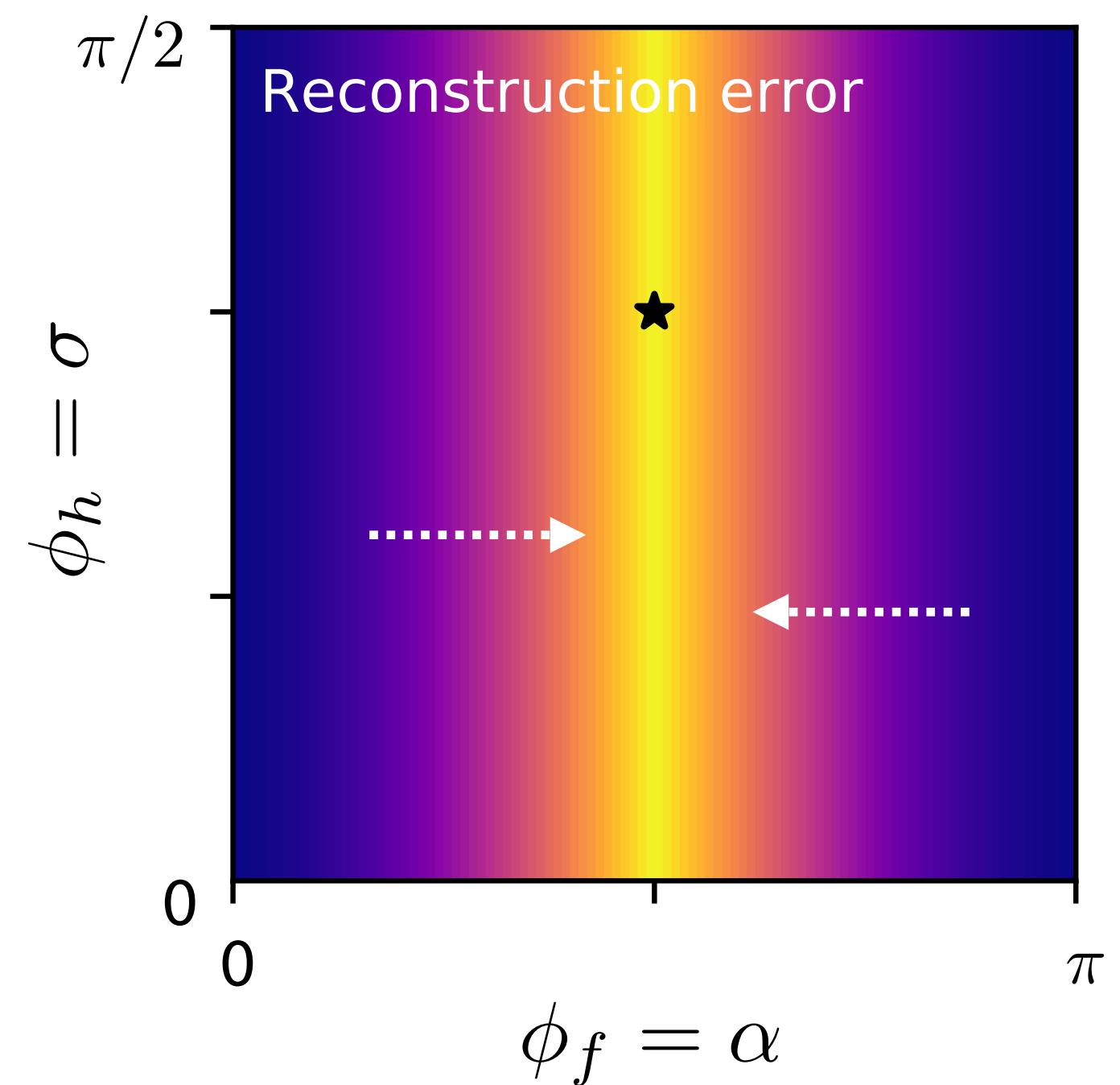
⇒ Learning ϕ_f by maximum
likelihood is unstable



M/D training

Solution: separate training in two phases!

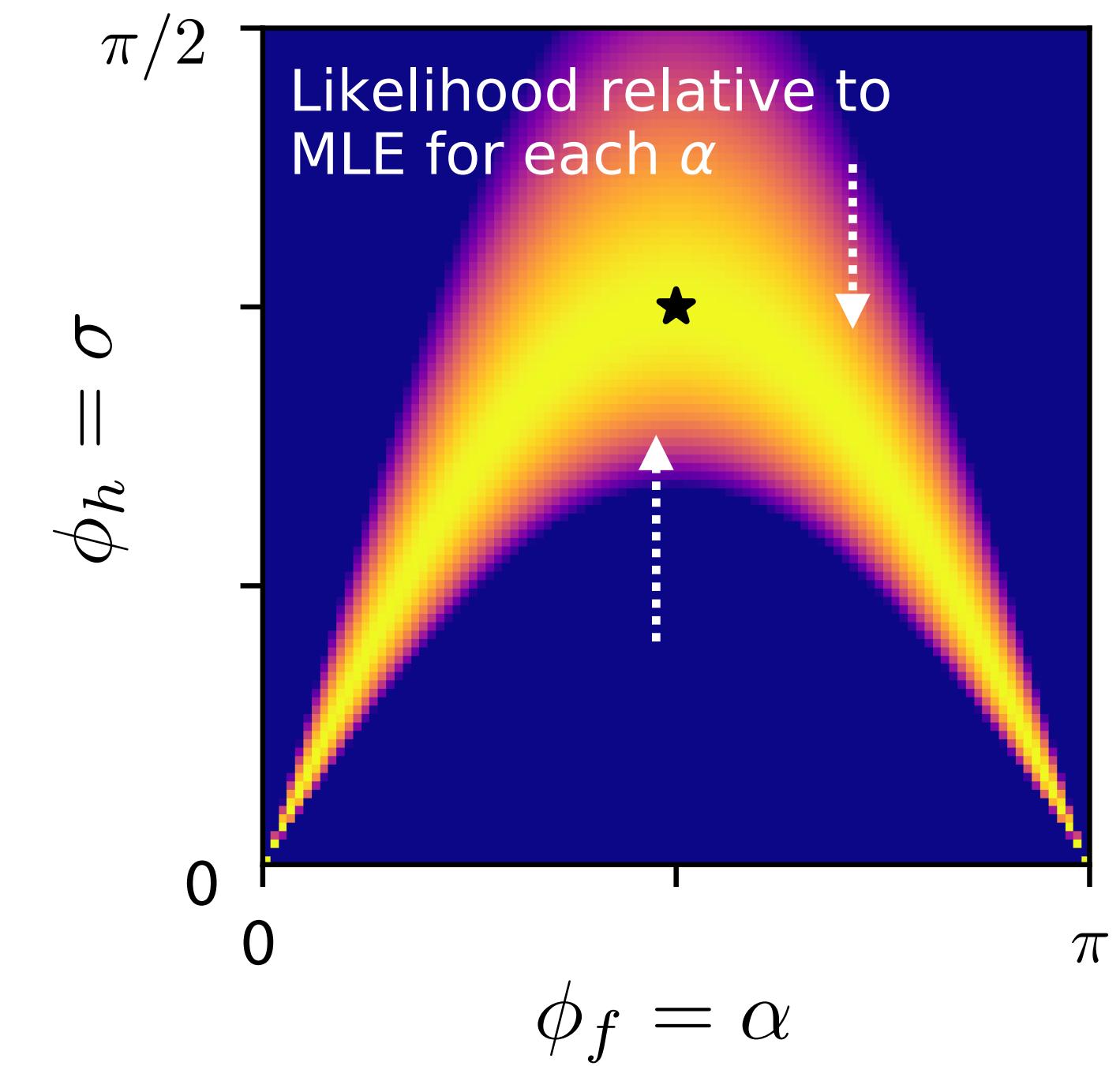
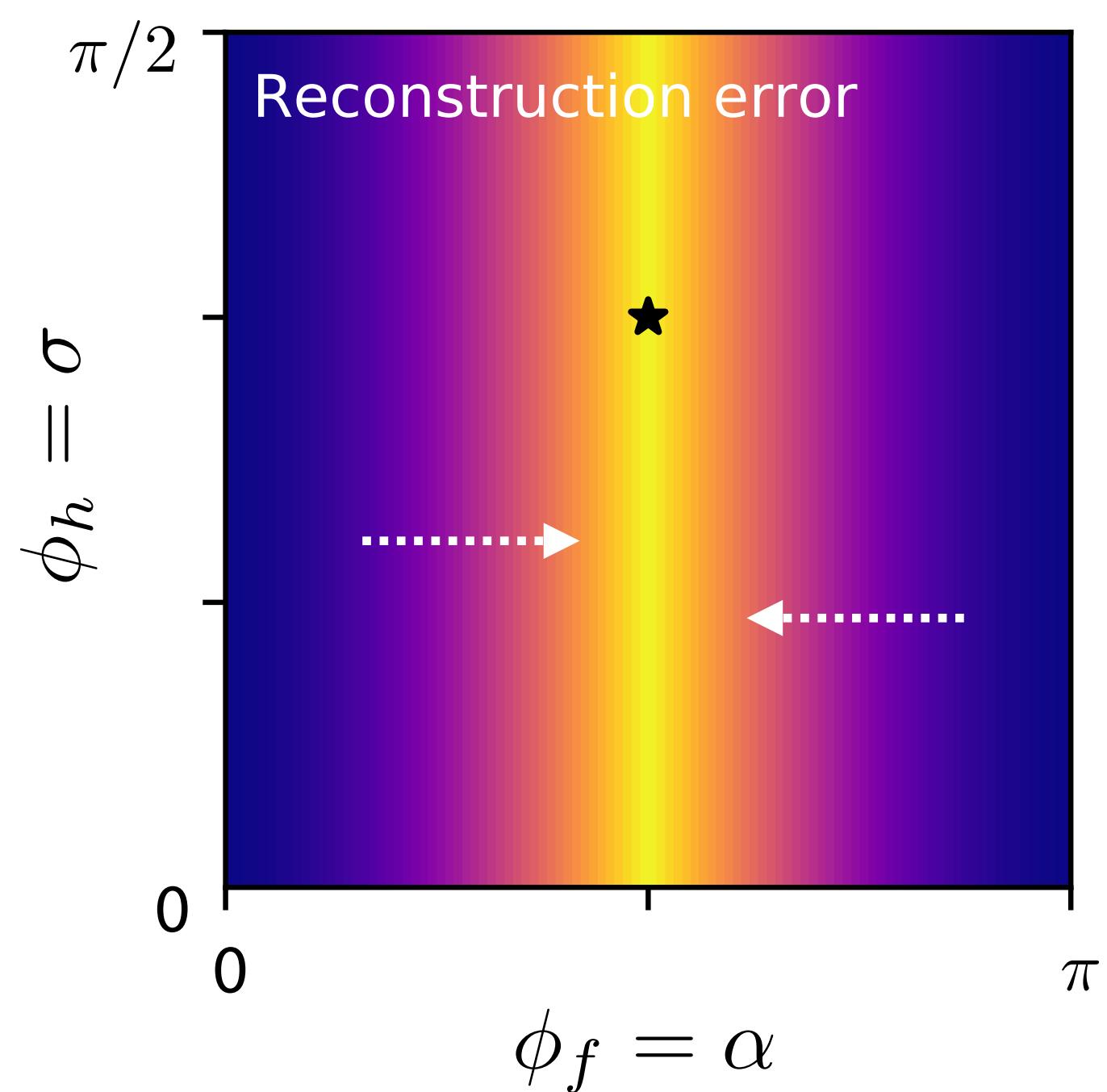
- **Manifold phase:**
update ϕ_f (and thus \mathcal{M}) by minimizing $\|x - x'\|$



M/D training

Solution: separate training in two phases!

- **Manifold phase:**
update ϕ_f (and thus \mathcal{M}) by minimizing $\|x - x'\|$
- **Density phase:**
update ϕ_h (and thus $p_{\mathcal{M}}(x)$) by maximum likelihood
(keeping \mathcal{M} fixed)



A second problem... and an accidental solution

The likelihood becomes expensive to evaluate for high-dimensional x :

$$\log p_{\mathcal{M}}(x) = \log p_{\tilde{u}}(h^{-1}(u)) - \log \det J_h(h^{-1}(u)) - \frac{1}{2} \log \det \left[(\mathbb{1} \ 0) J_f^T(u) J_f(u) \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right]$$

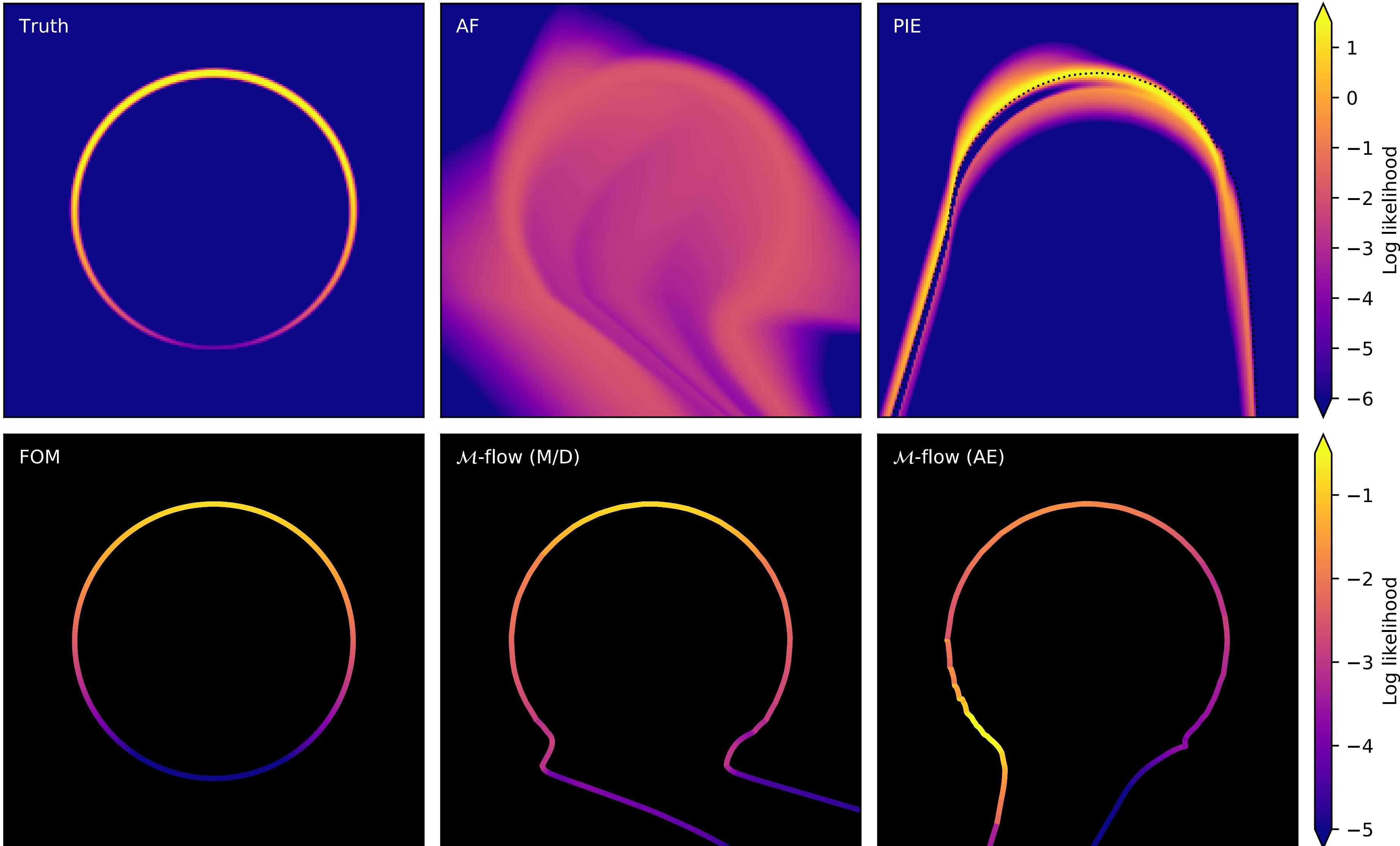
Cannot separate determinant of product of non-square matrices

M/D training sidesteps this problem: density phase only requires gradient

$$\nabla_{\phi_h} (\log p_{\mathcal{M}}(x)) = \nabla_{\phi_h} (\log p_{\tilde{u}}(h^{-1}(u))) - \nabla_{\phi_h} (\log \det J_h(h^{-1}(u))) - \underbrace{\nabla_{\phi_h} \frac{1}{2} \log \det \left[(\mathbb{1} \ 0) J_f^T(u) J_f(u) \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right]}_{=0},$$

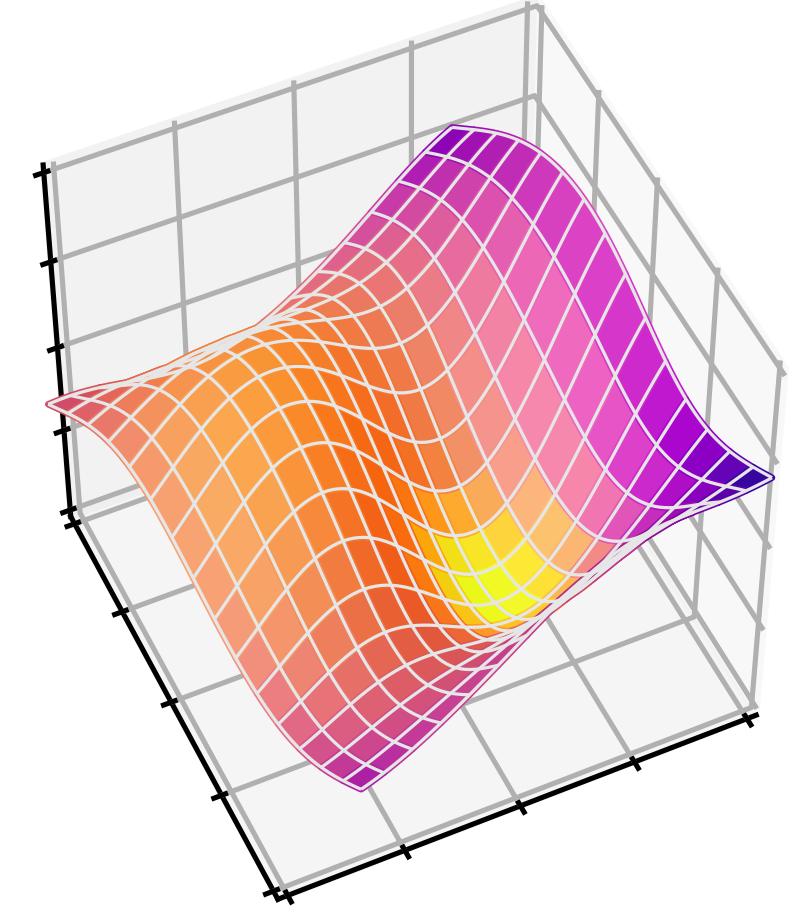
which can be computed efficiently!

Gaussian on a circle

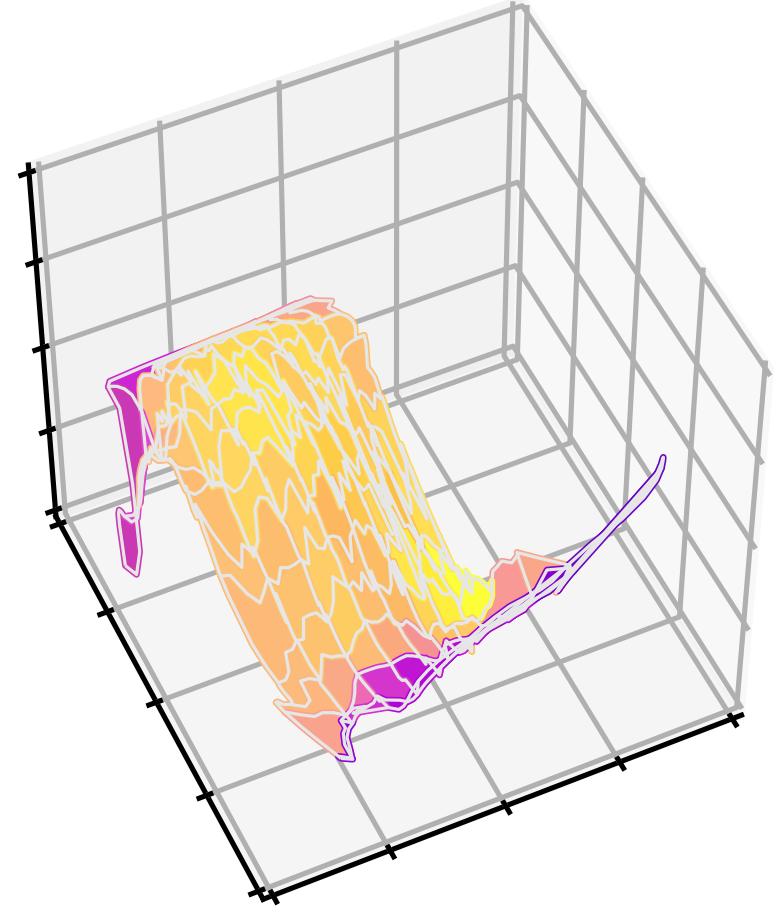


Mixture model on a polynomial surface

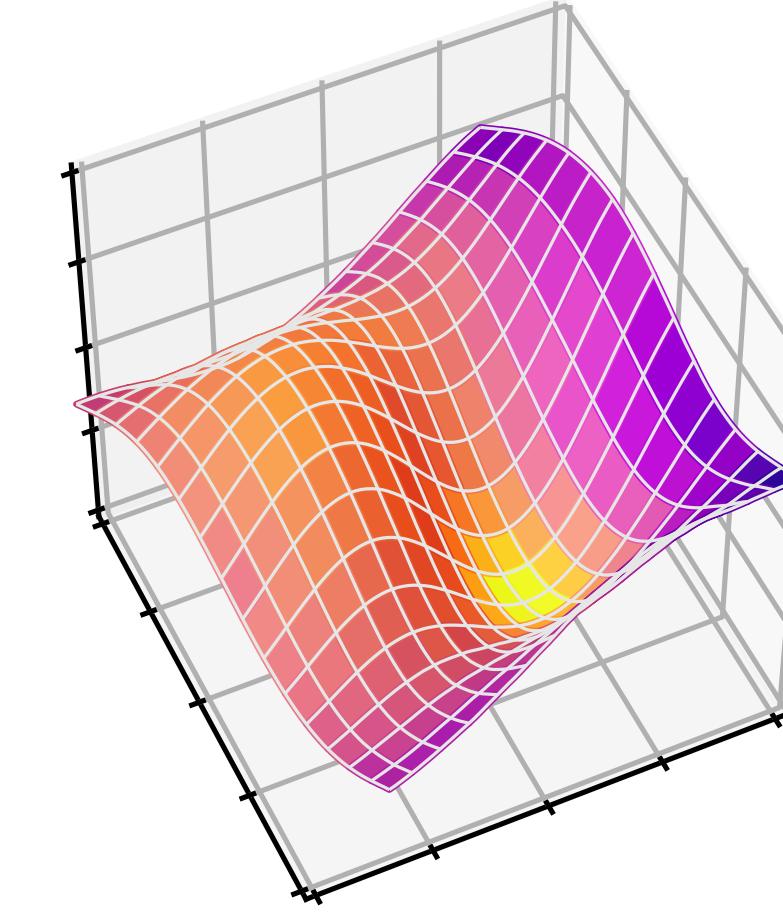
Ground truth, $\theta = 0$



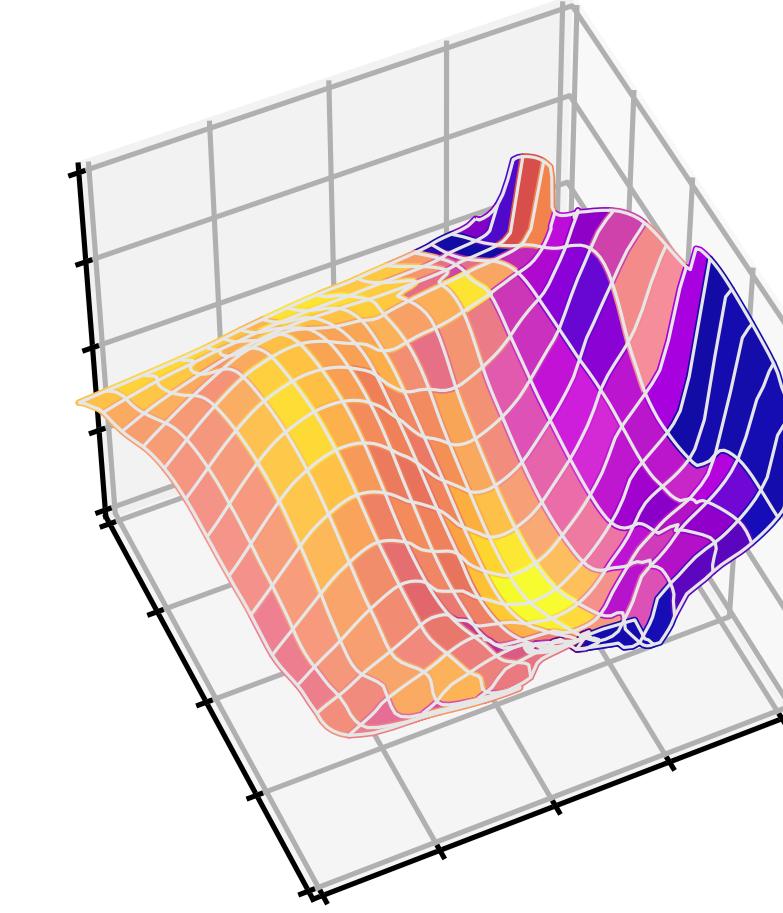
PIE, $\theta = 0$



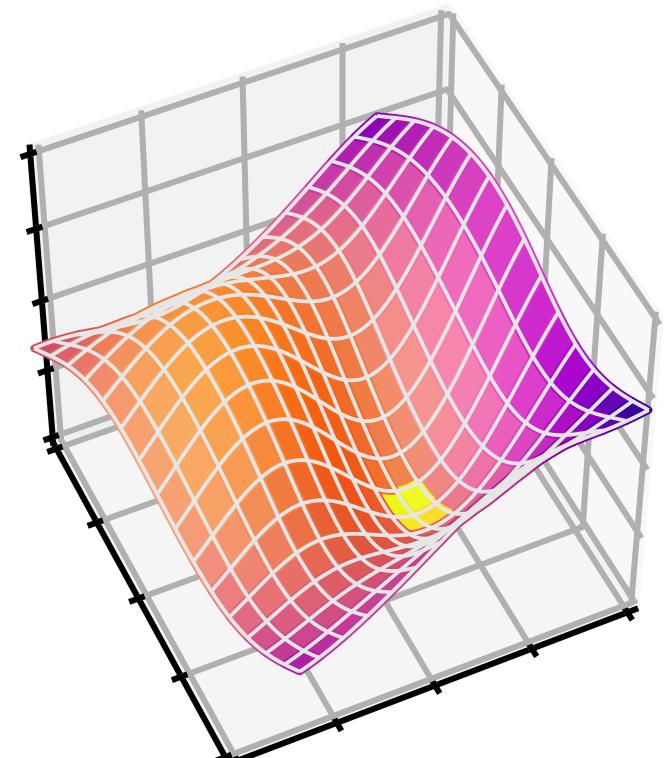
\mathcal{M} -flow (M/D), $\theta = 0$



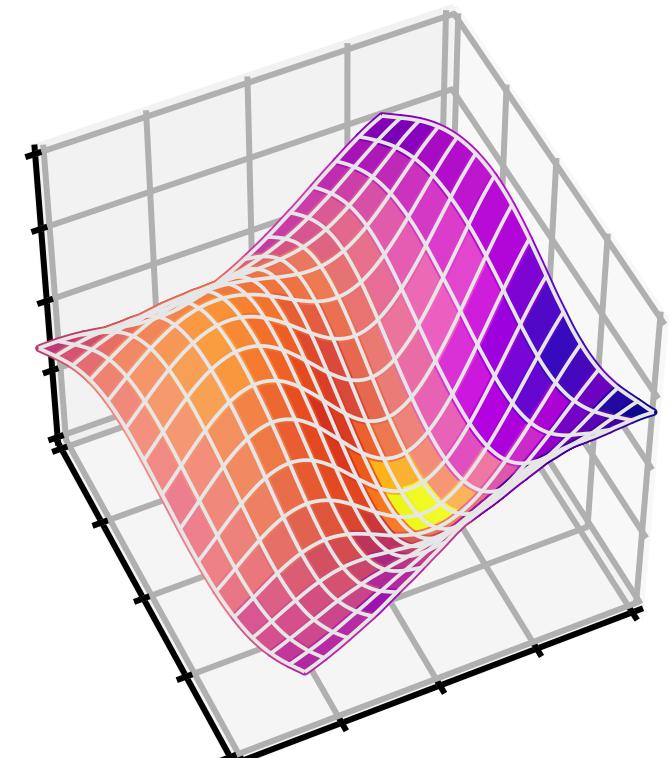
\mathcal{M} -flow (OT), $\theta = 0$



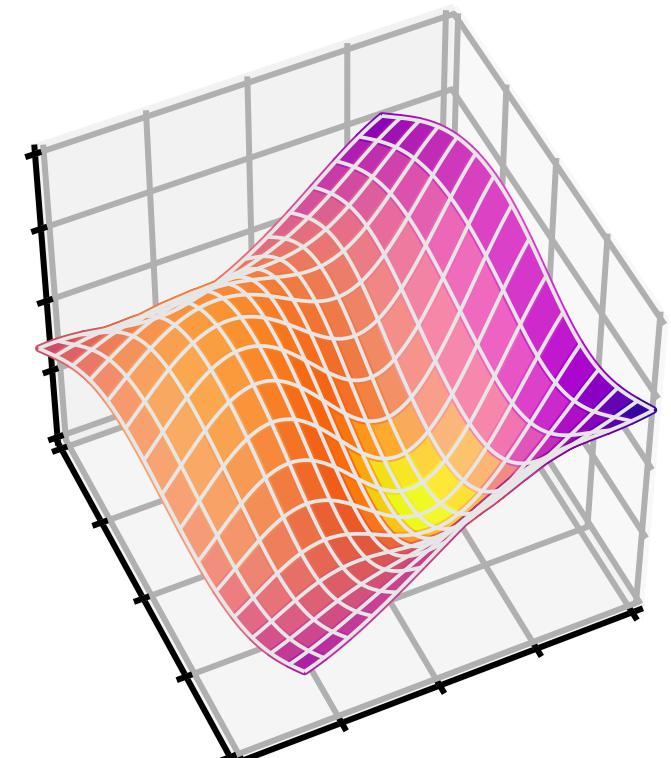
Ground truth, $\theta = -1$



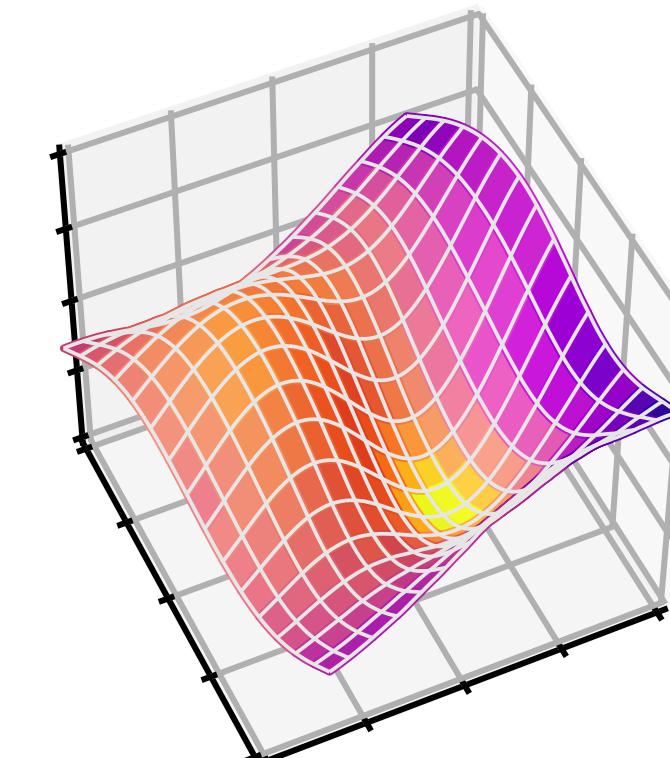
\mathcal{M} -flow, $\theta = -1$



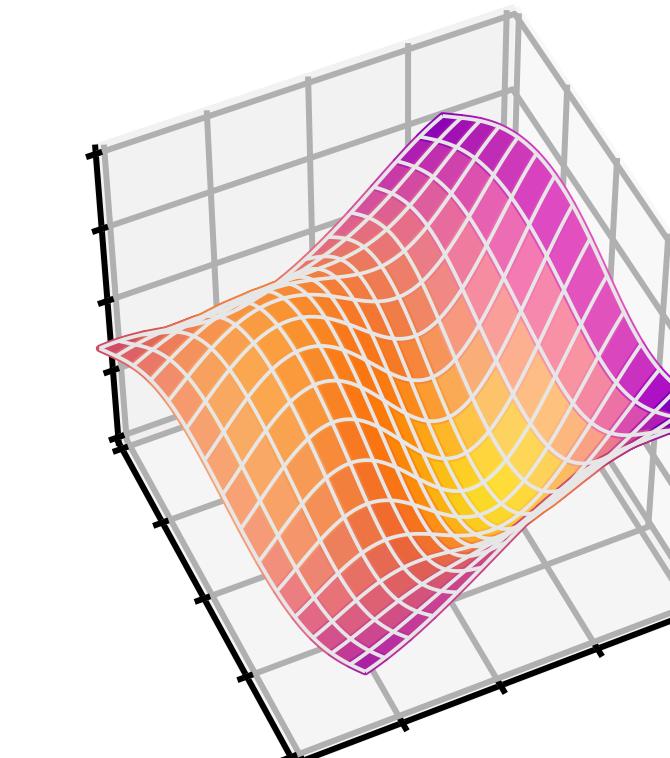
Ground truth, $\theta = 0$



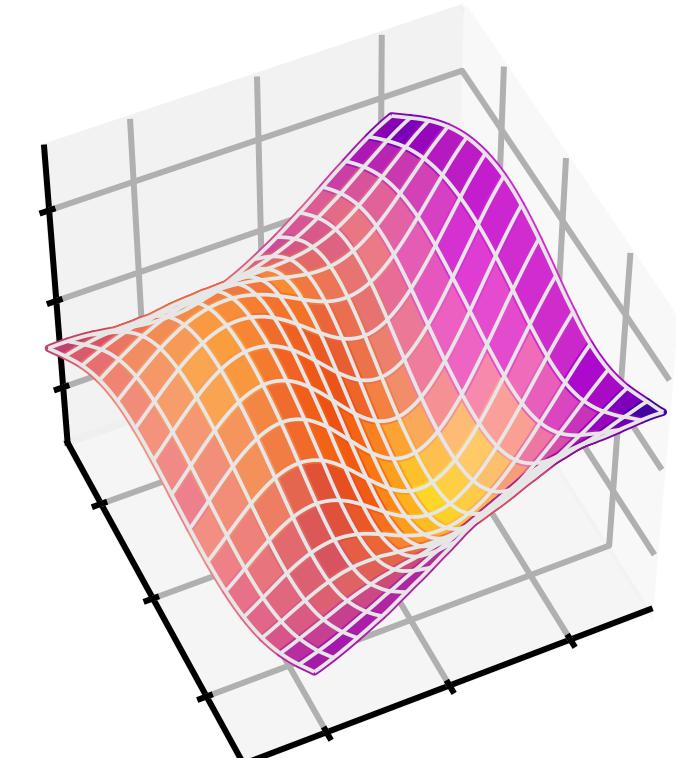
\mathcal{M} -flow, $\theta = 0$



Ground truth, $\theta = 1$

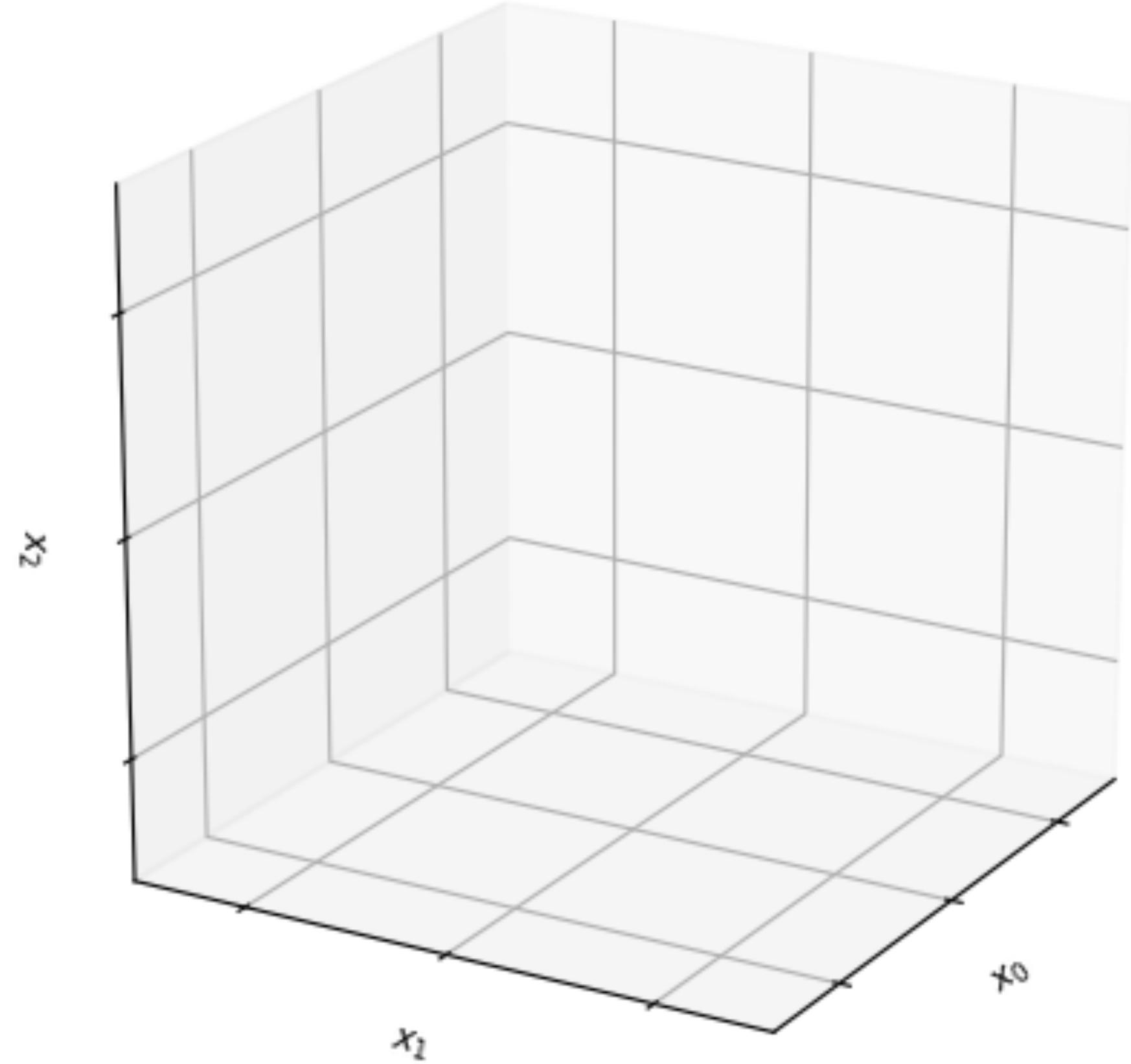


\mathcal{M} -flow, $\theta = 1$



Lorenz attractor

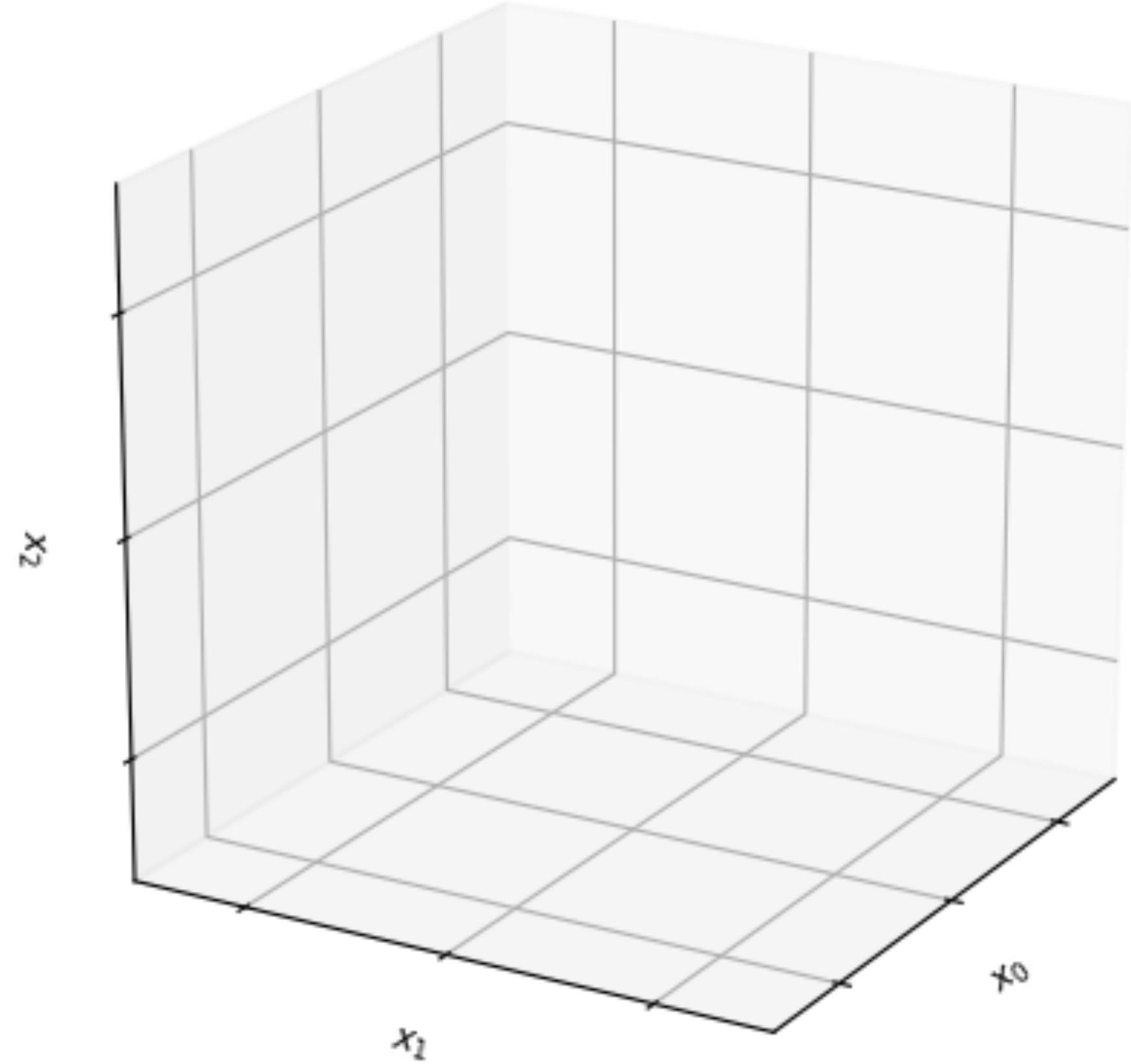
[E. Lorenz 1963]



$$\frac{dx_0}{dt} = \sigma(x_1 - x_0), \quad \frac{dx_1}{dt} = x_0(\rho - x_2) - x_1, \quad \frac{dx_2}{dt} = x_0x_1 - \beta x_2.$$

Lorenz attractor

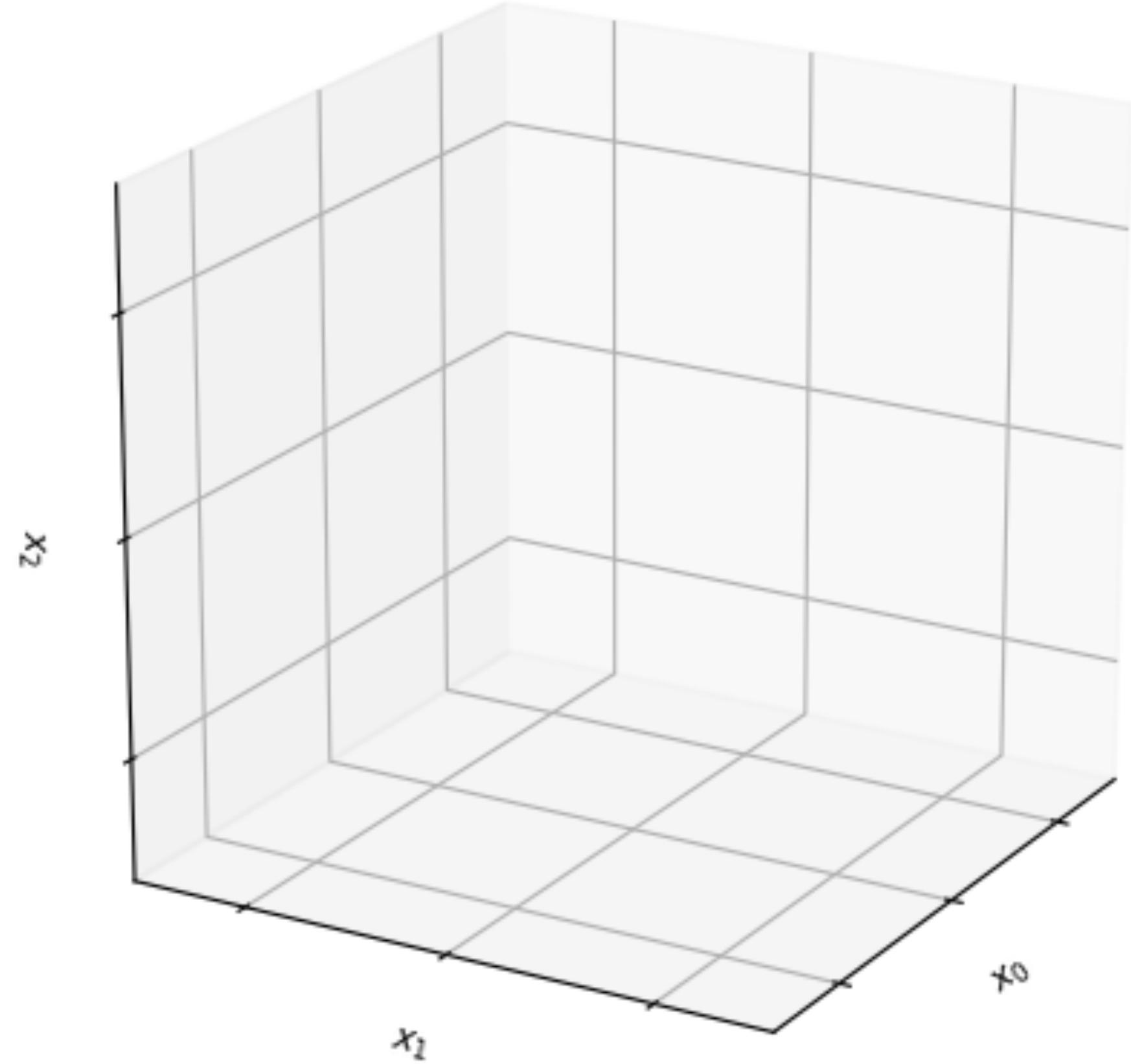
[E. Lorenz 1963]



$$\frac{dx_0}{dt} = \sigma(x_1 - x_0), \quad \frac{dx_1}{dt} = x_0(\rho - x_2) - x_1, \quad \frac{dx_2}{dt} = x_0x_1 - \beta x_2.$$

Lorenz attractor

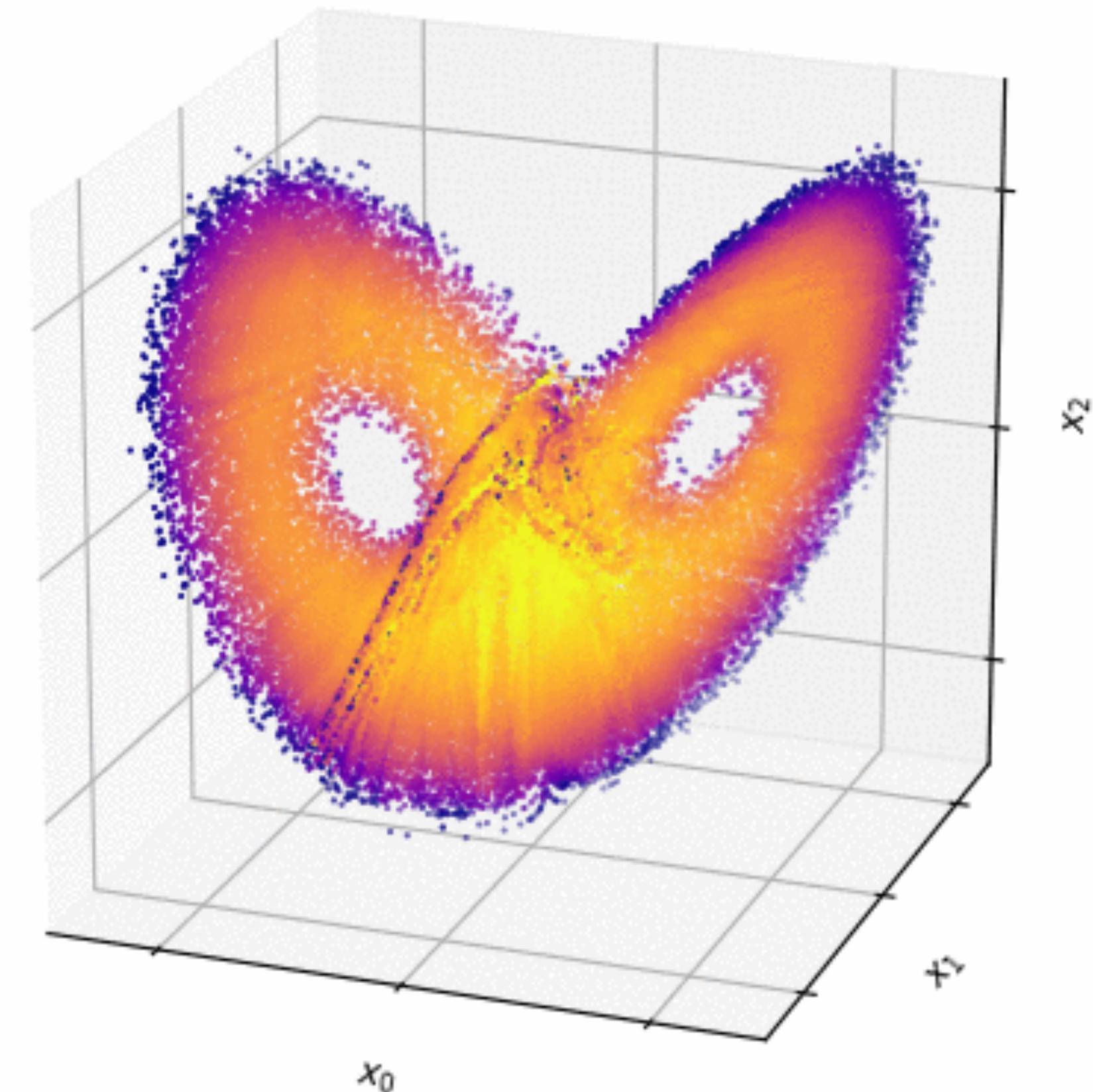
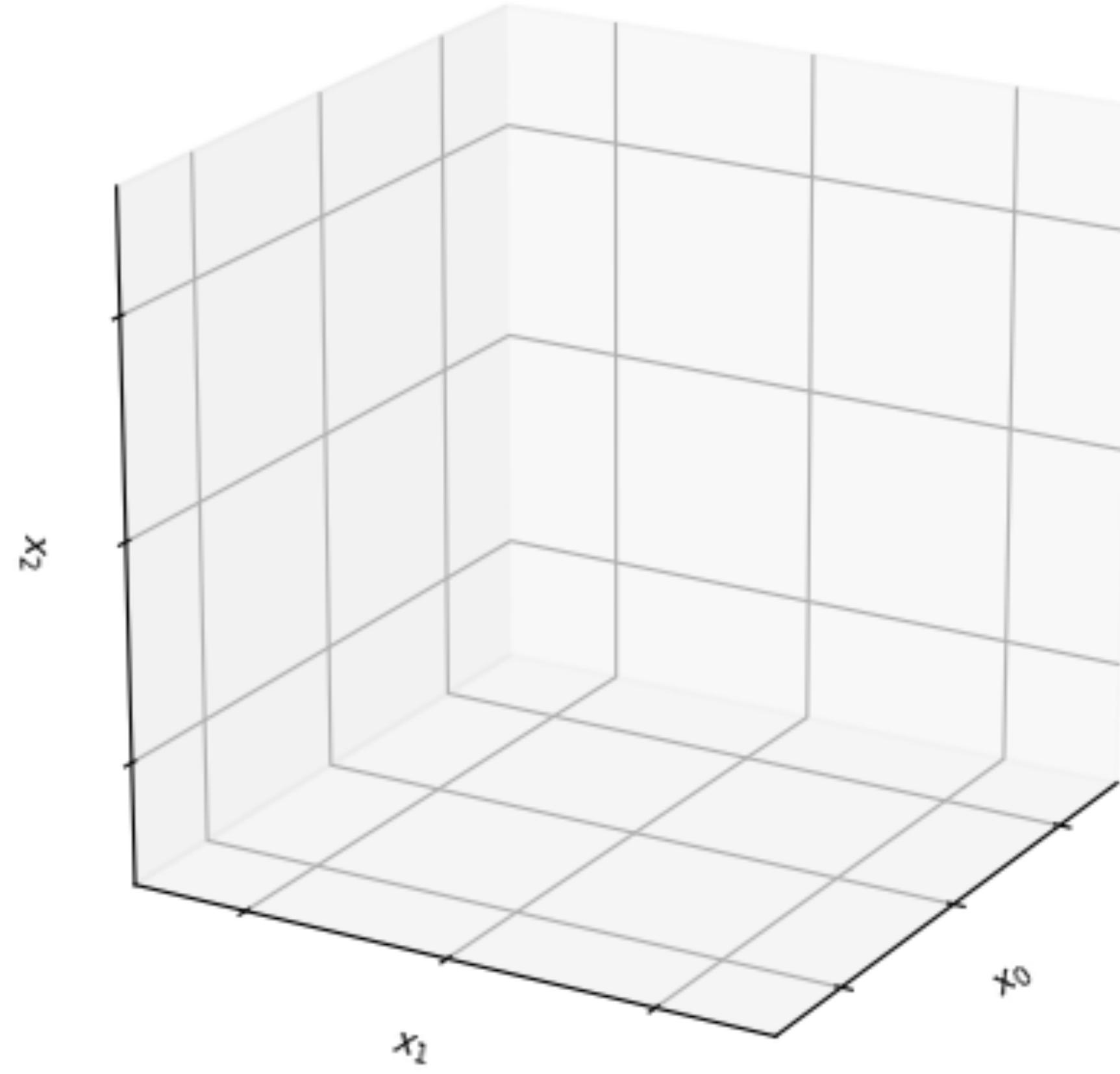
[E. Lorenz 1963]



$$\frac{dx_0}{dt} = \sigma(x_1 - x_0), \quad \frac{dx_1}{dt} = x_0(\rho - x_2) - x_1, \quad \frac{dx_2}{dt} = x_0x_1 - \beta x_2.$$

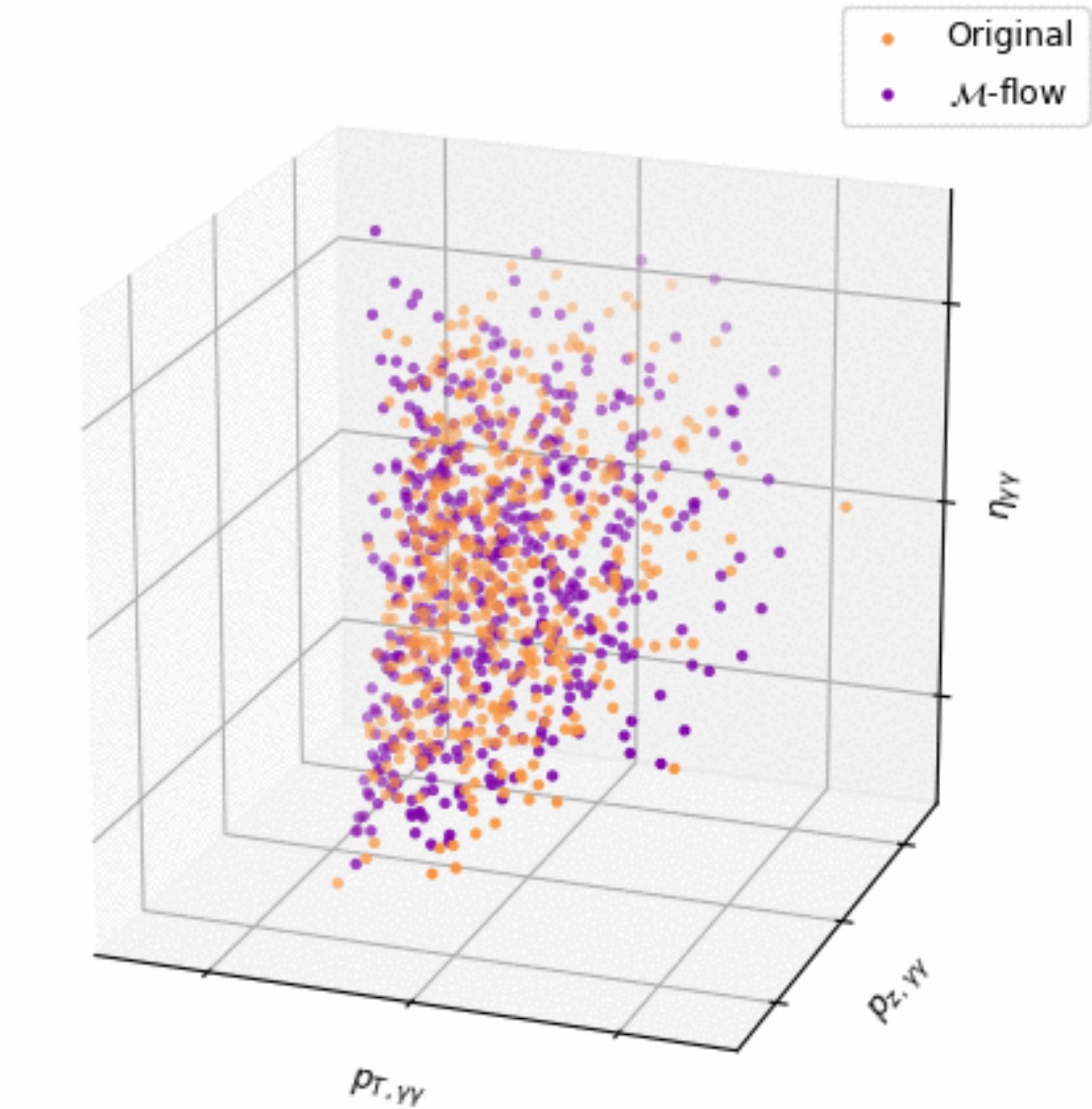
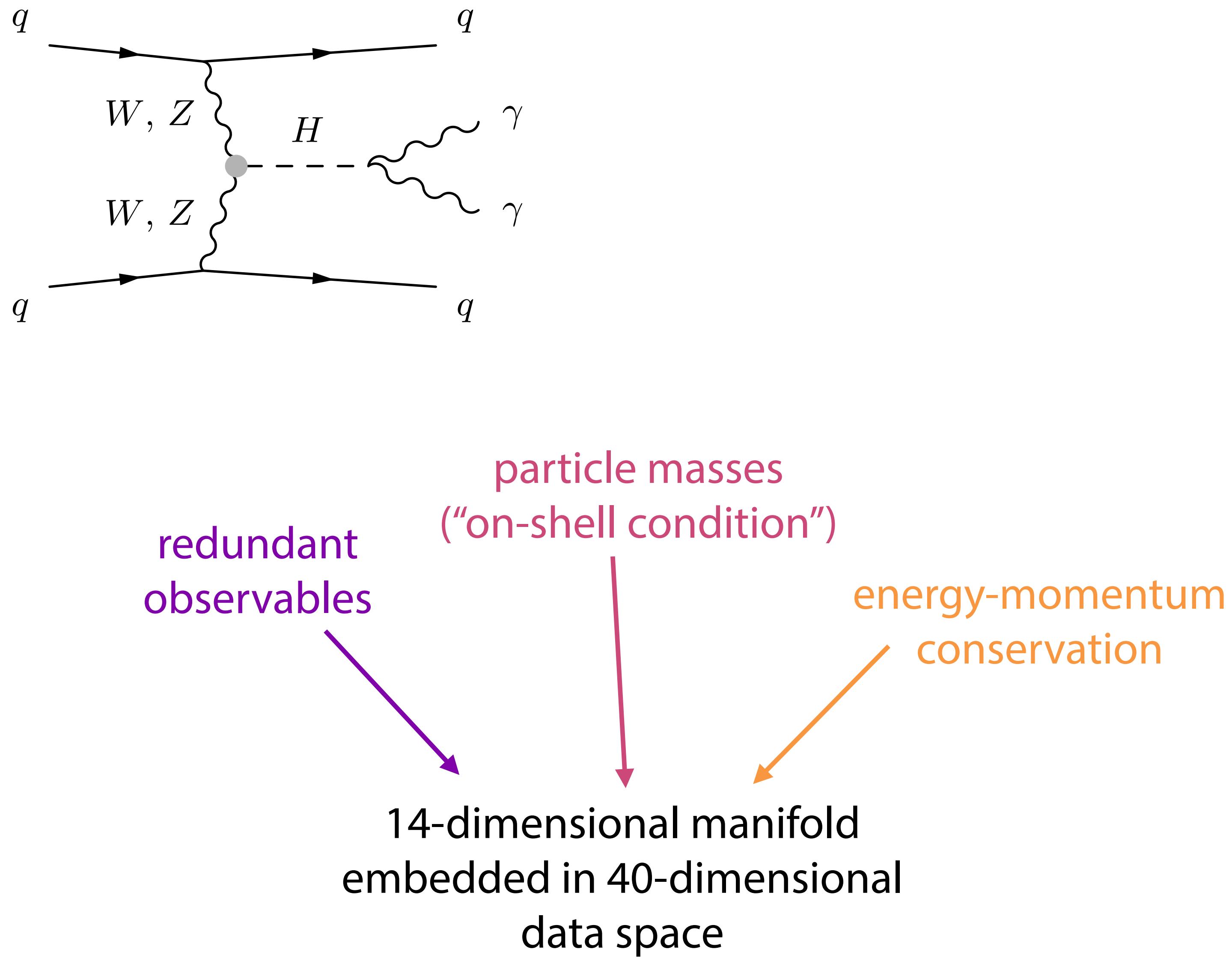
Lorenz attractor

[E. Lorenz 1963]

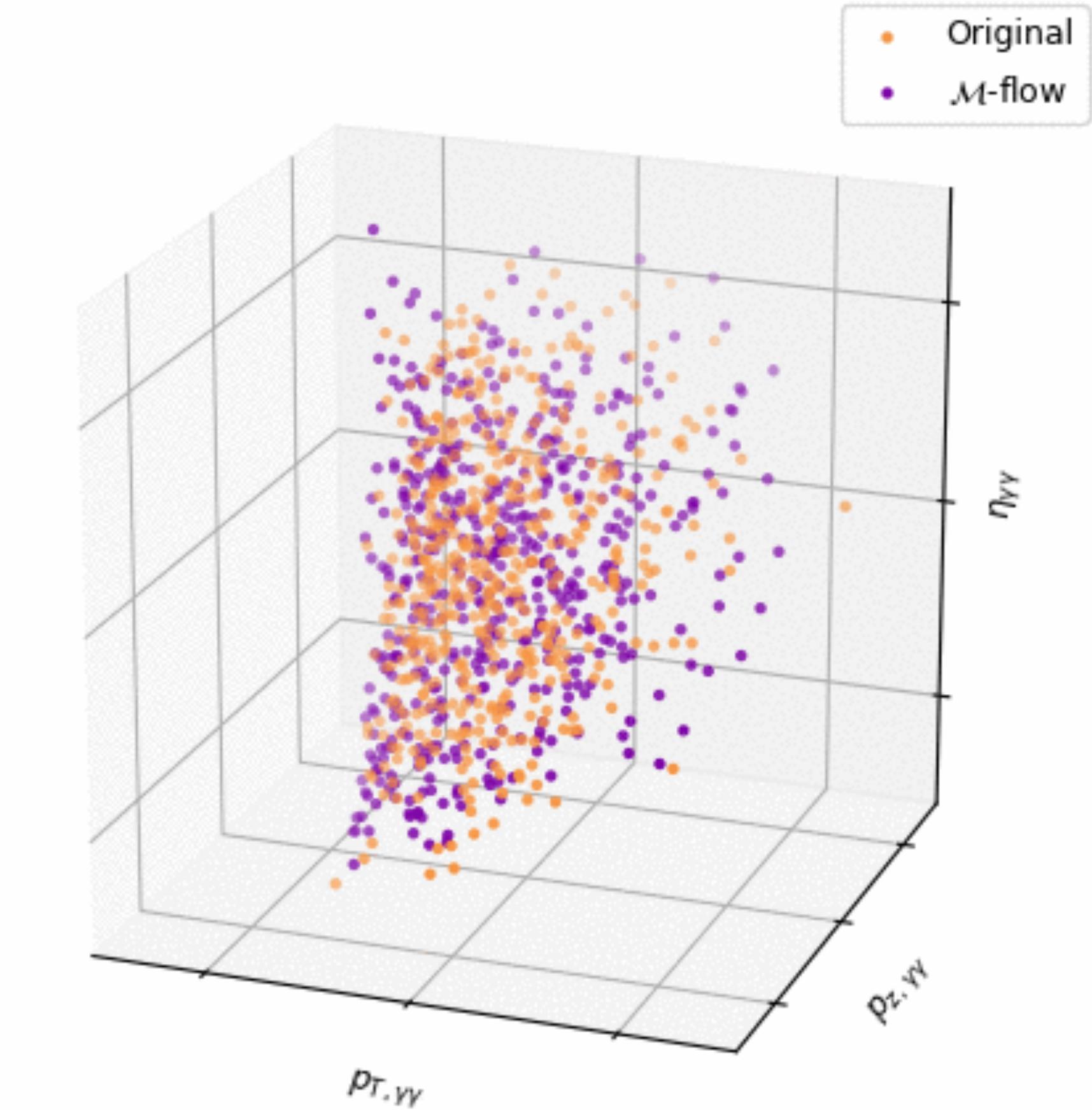
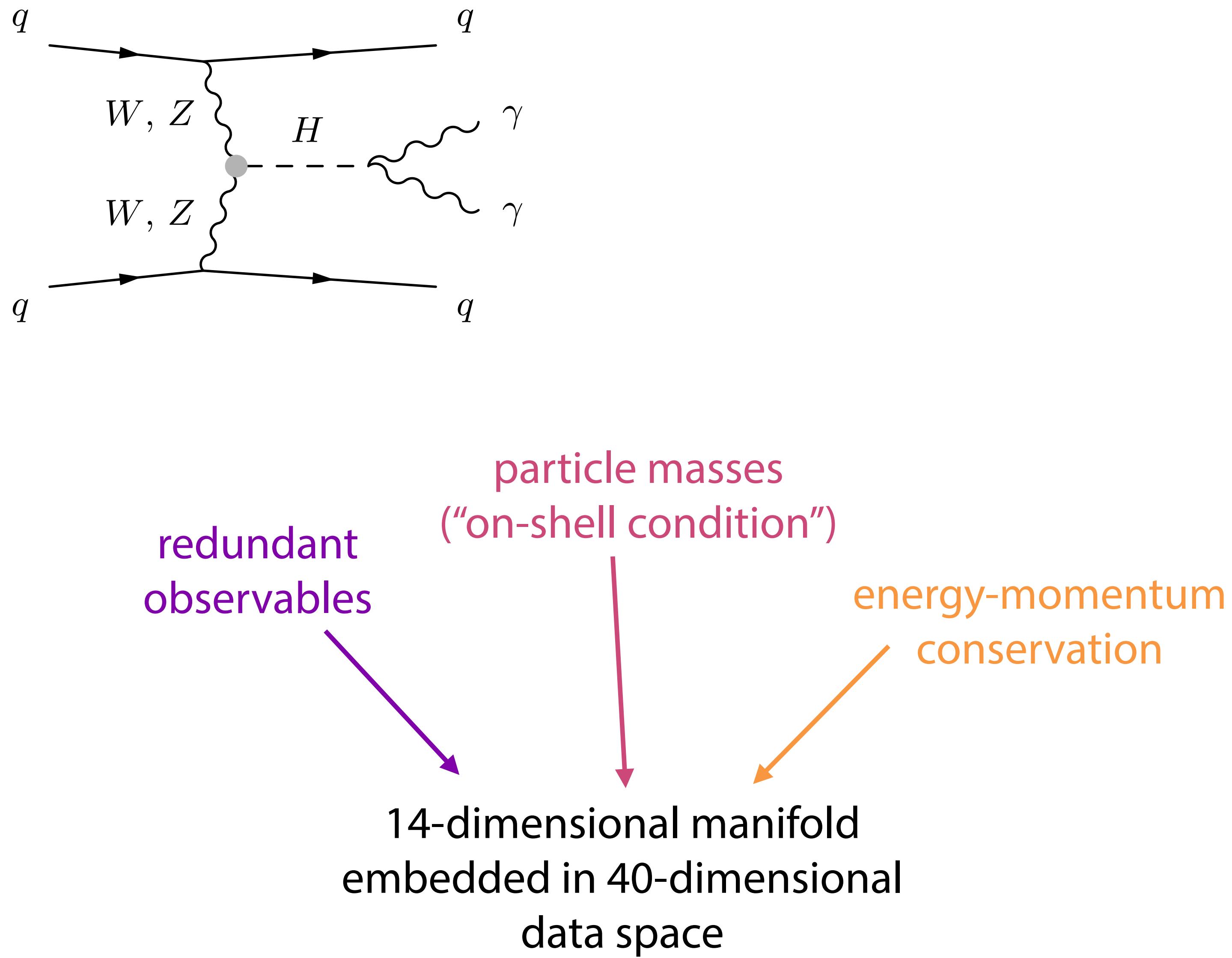


$$\frac{dx_0}{dt} = \sigma(x_1 - x_0), \quad \frac{dx_1}{dt} = x_0(\rho - x_2) - x_1, \quad \frac{dx_2}{dt} = x_0x_1 - \beta x_2.$$

Particle physics: structure



Particle physics: structure



Particle physics: results

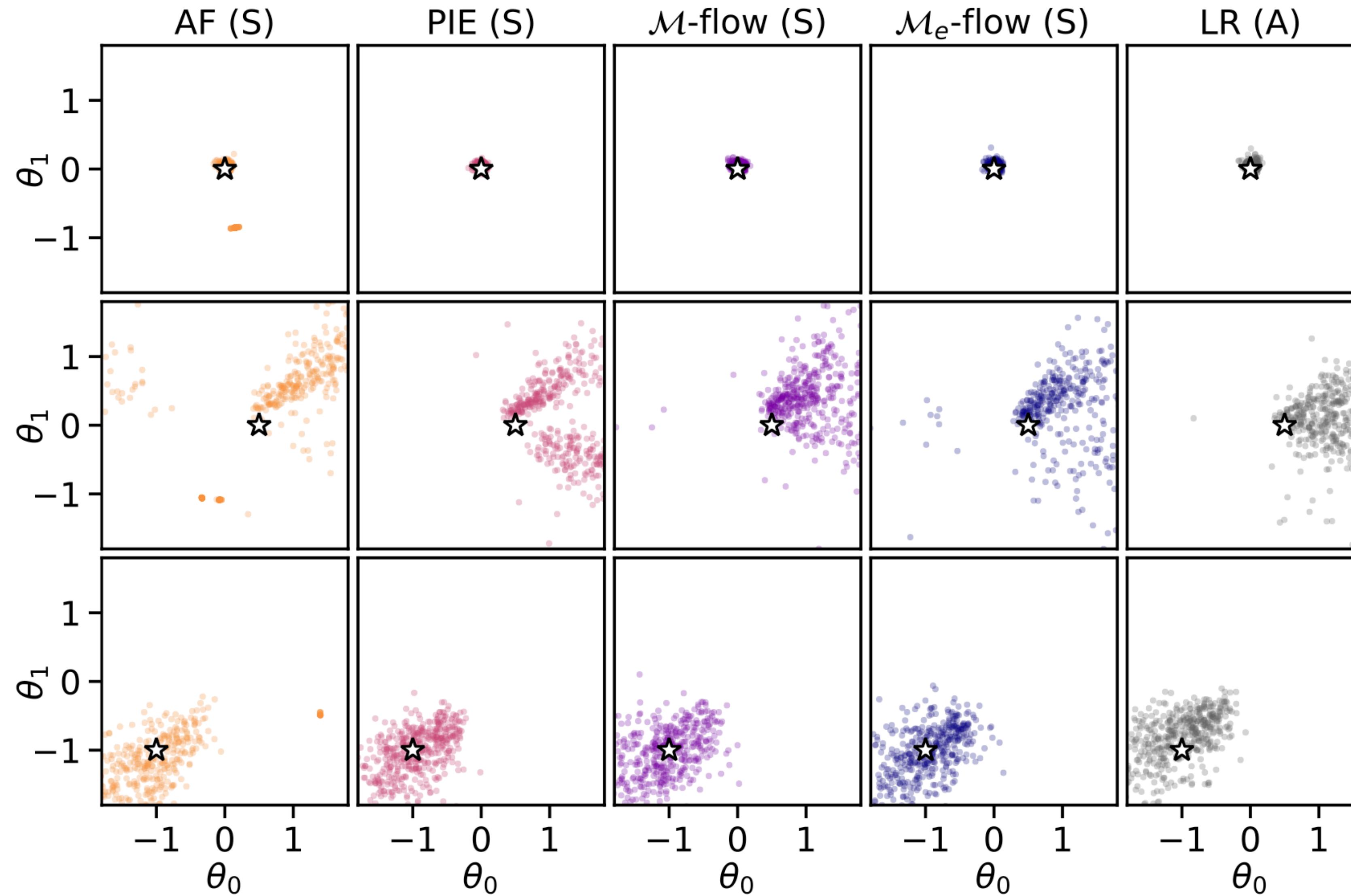
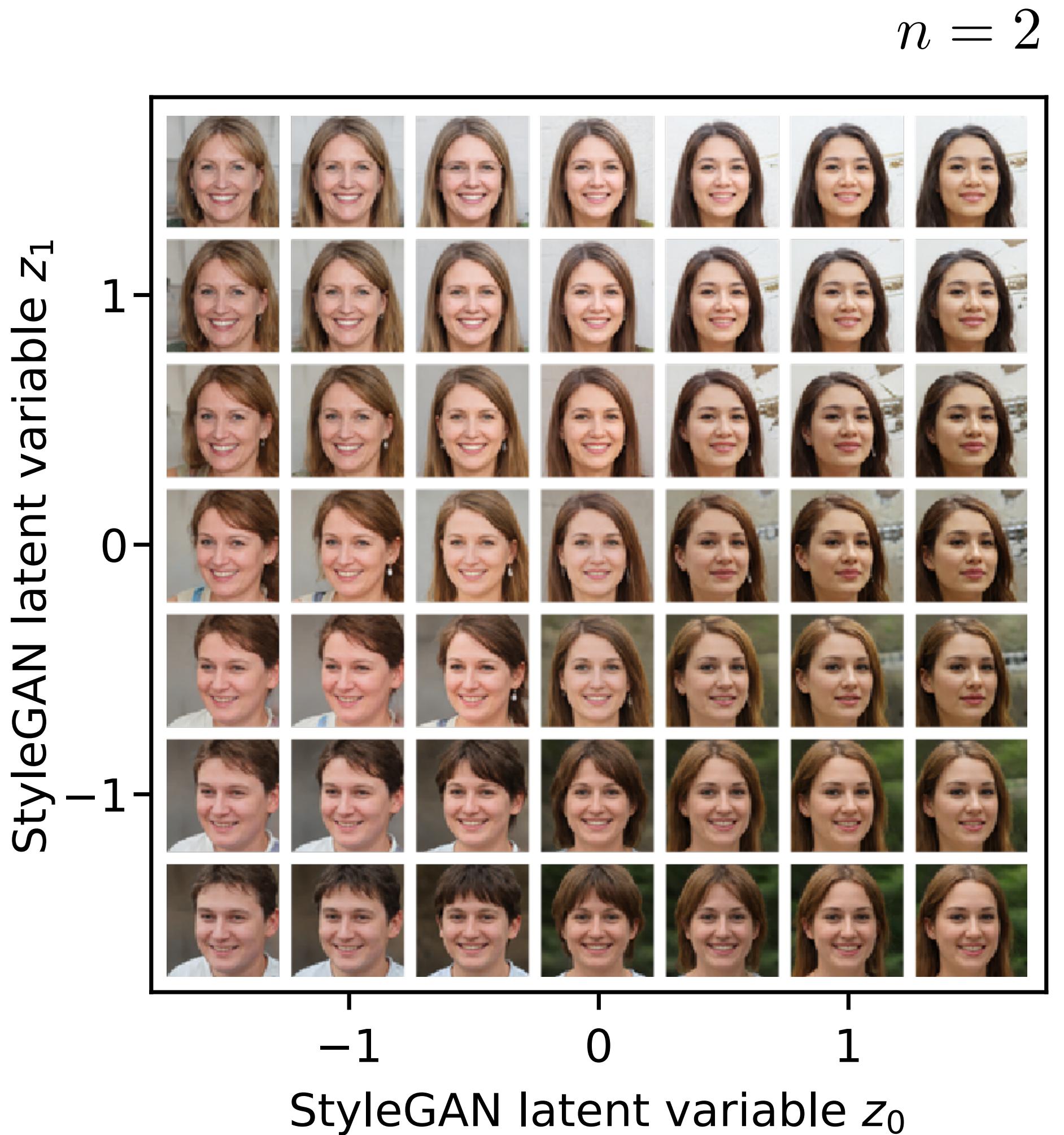


Image manifolds

Q: How to make image datasets where we **know** that data lives on an n -dimensional manifold?

A: take a pretrained GAN model, sample n of its latent variables, and keep all others fixed



Samples

Test data



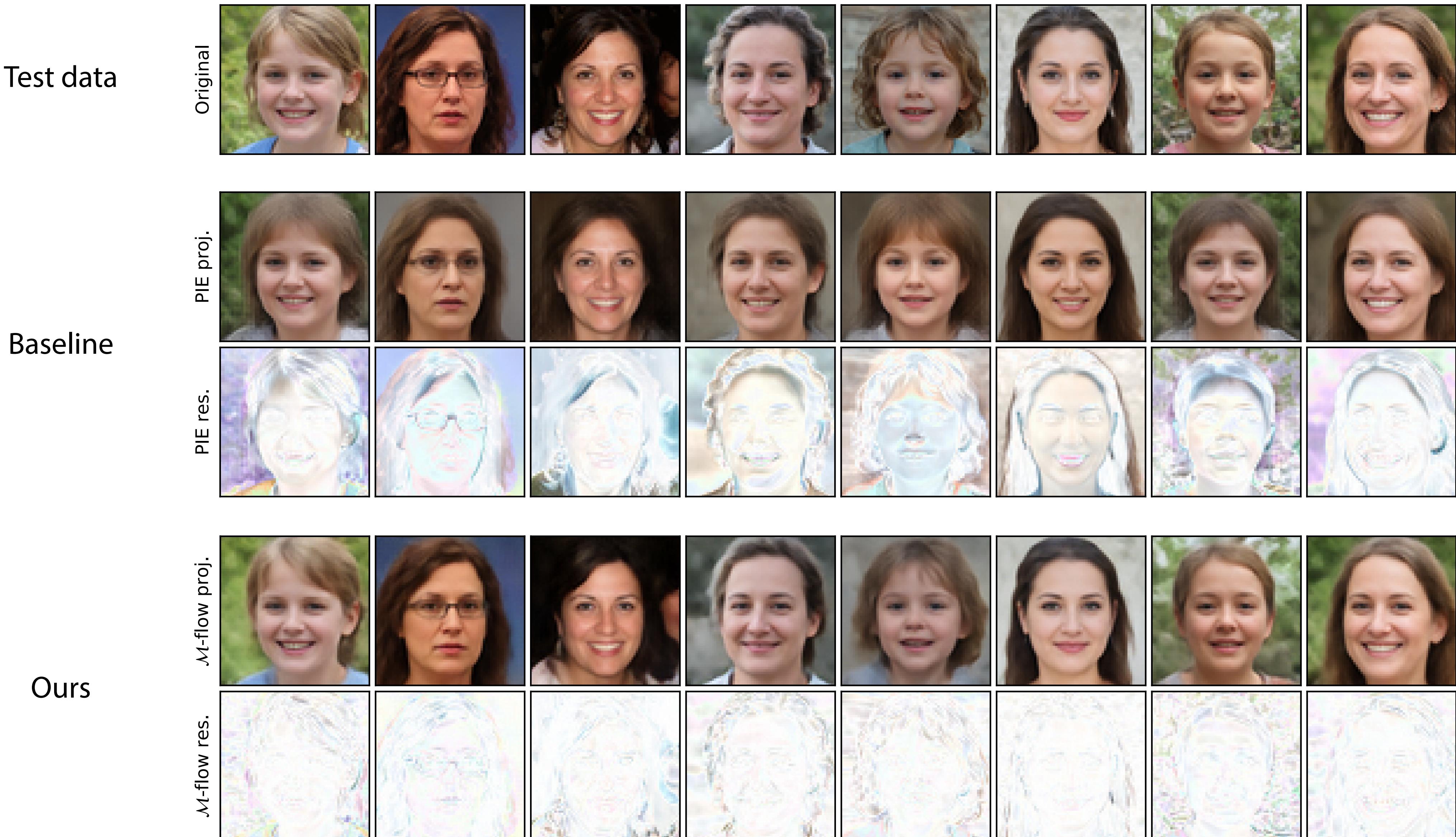
Baselines



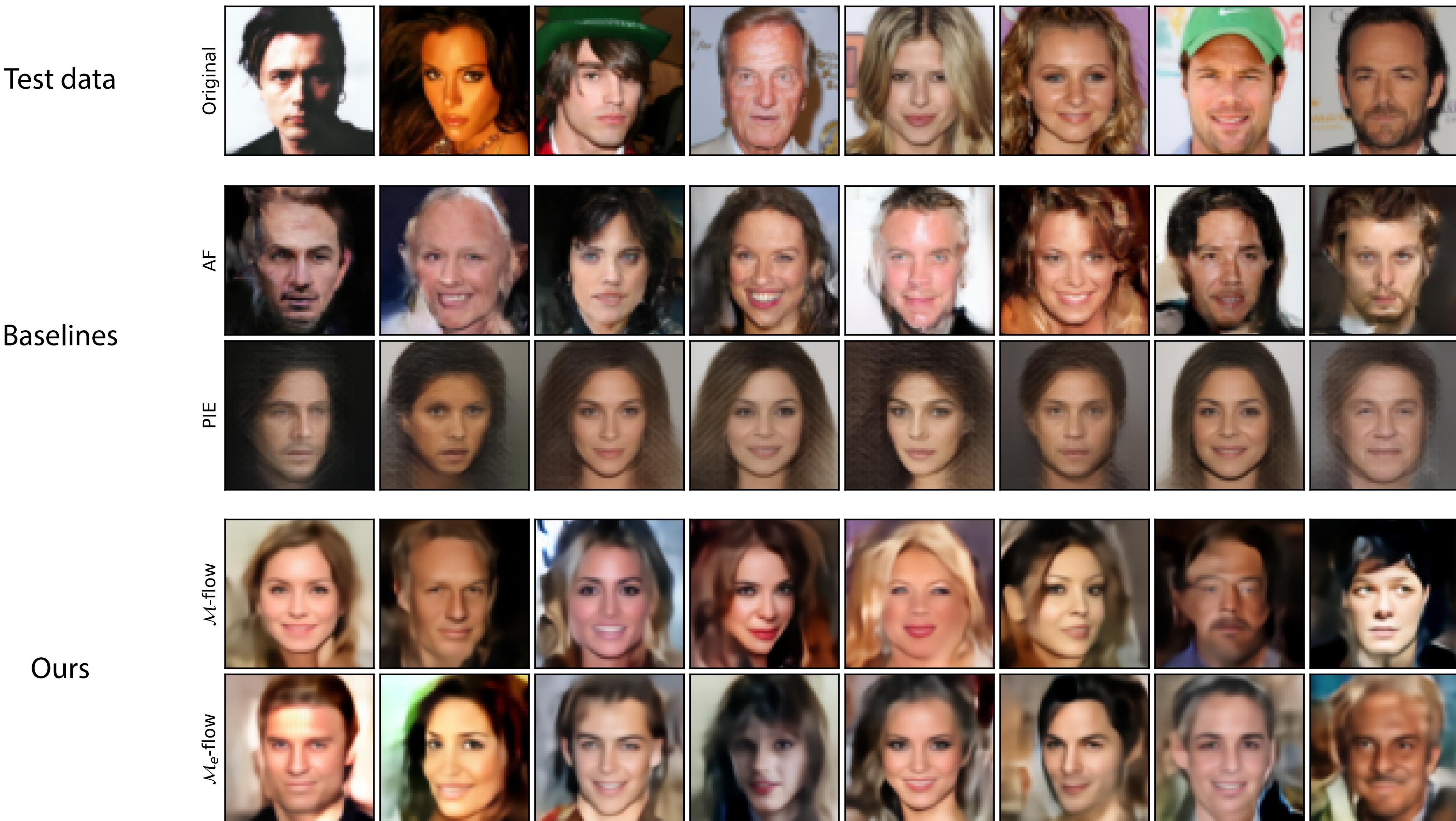
Ours



Projections to learned manifolds



Real-world images: CelebA samples



CelebA projections

Test data



Baseline



Ours

