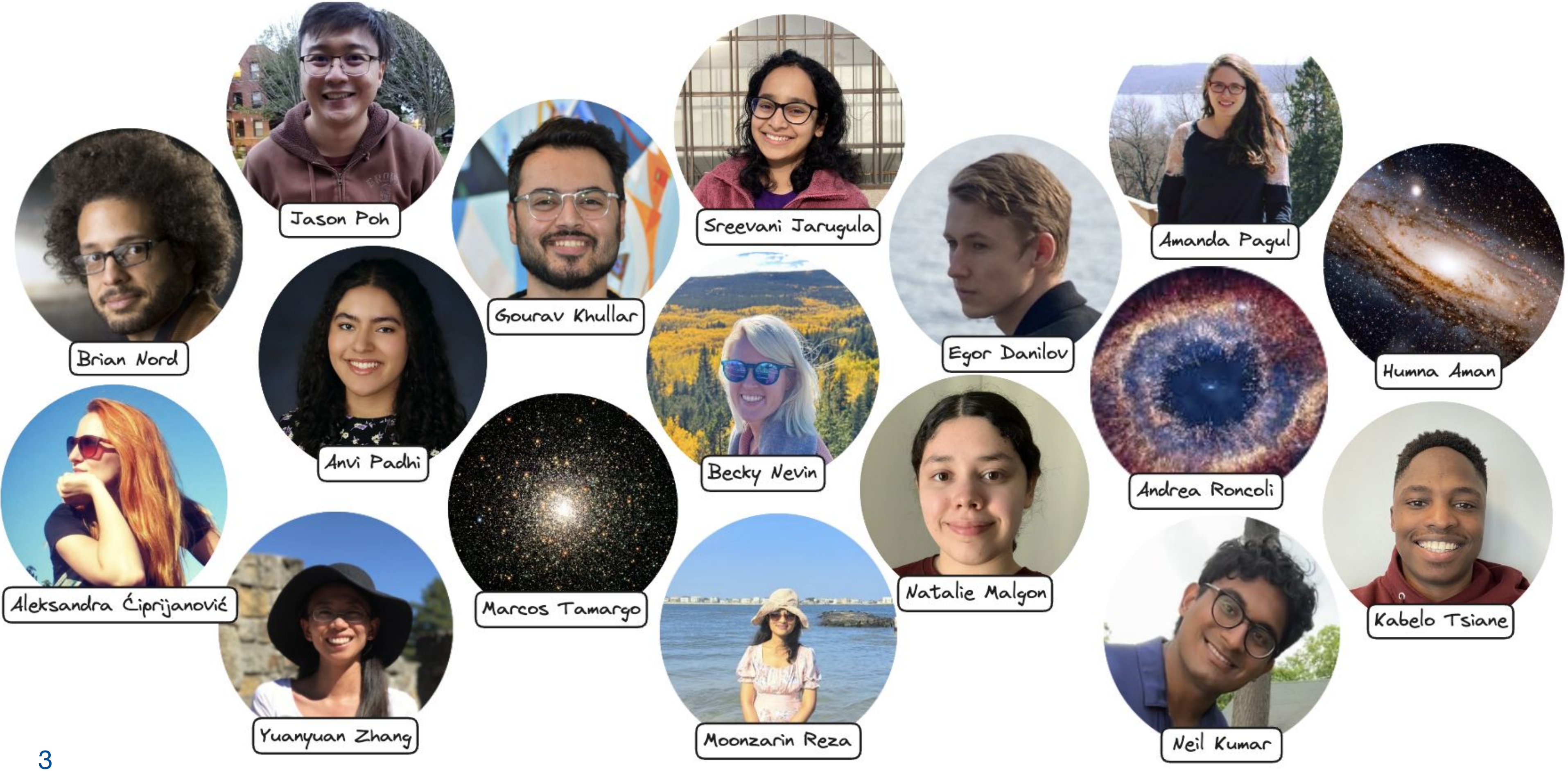
Simulation-based inference
[Deserved] hype, tutorial, and future outlook and challenges

Becky Nevin and DeepSkies Lab

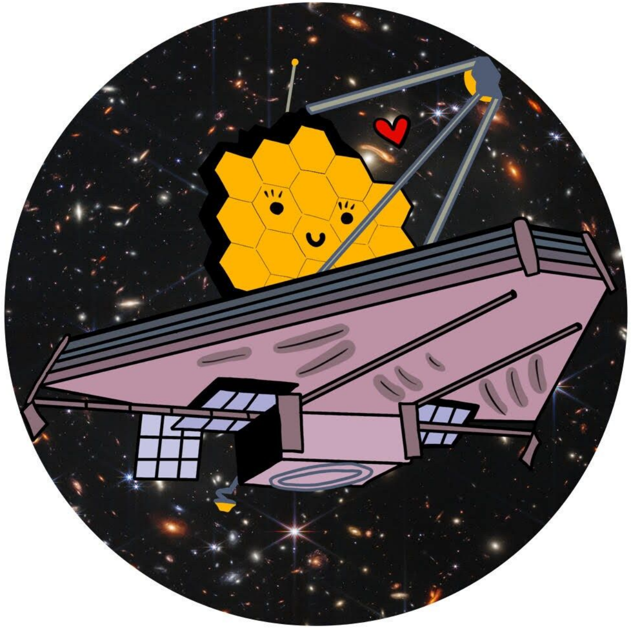
December, 2023
COFI AI/ML Winter School

About me

The Deepskies Lab SBI team



It’s Saturday, prizes!



Fermilab

Iteratively updating our presentation prior

Iteratively updating our presentation prior

This presentation is Bayesian

What do you know about simulation-based inference?

- Heard of it?
- Talked about it in a class?
- Done research with it?

Also sometimes called "likelihood-free inference" or "approximate bayesian computation"

What do you know about simulation-based inference?

- Heard of it?
- Talked about it in a class?
- Done research with it?

Also sometimes called "likelihood-free inference" or "approximate bayesian computation"

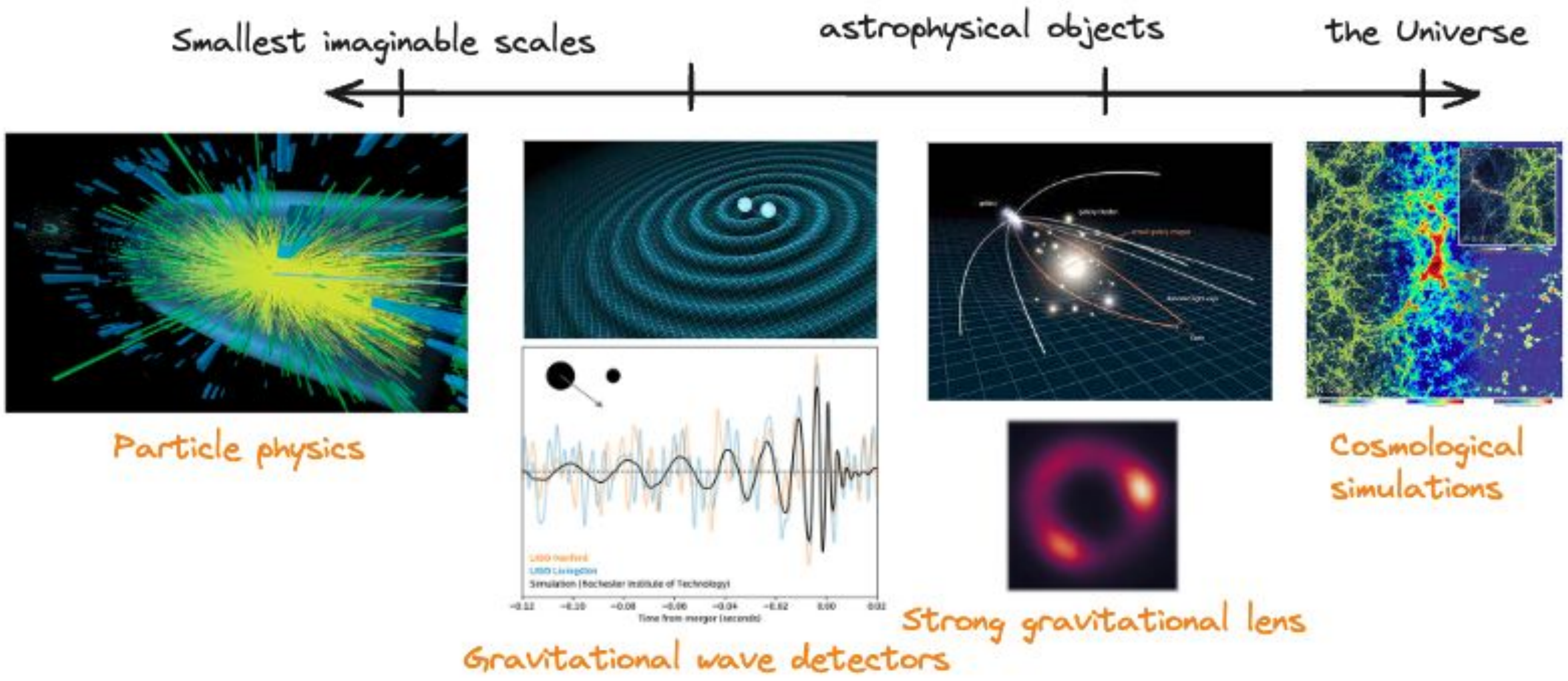
- Have you heard of likelihood-based inference? Ever run MCMC?

Visual outline

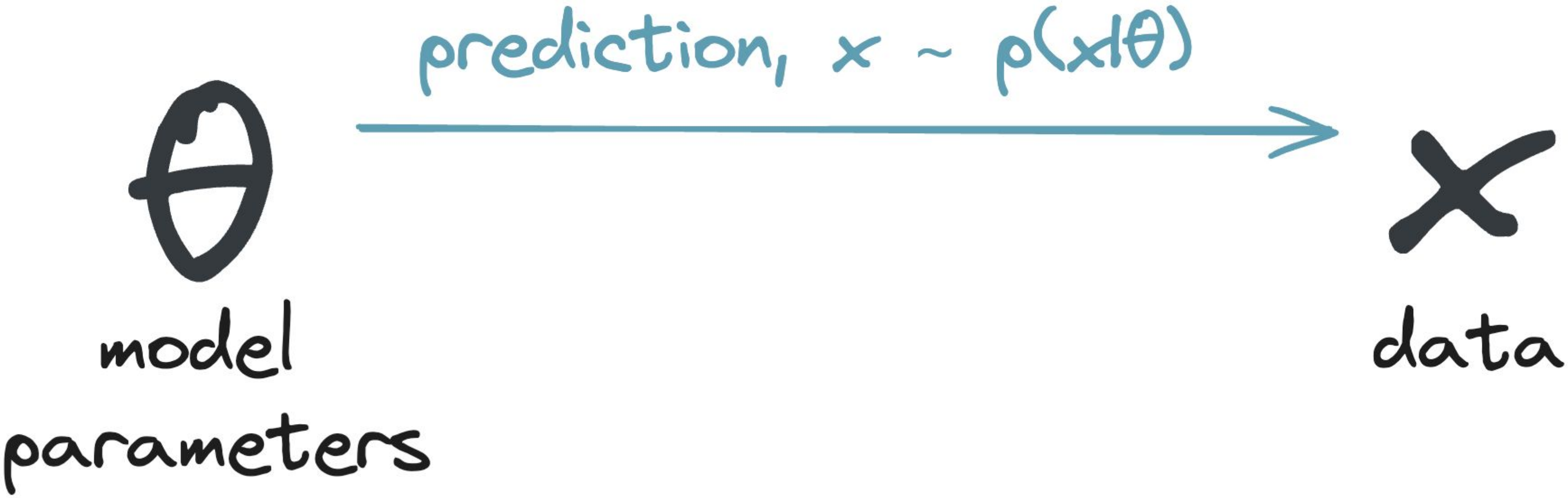
1. Simulation-based versus likelihood-based inference
2. Advantages of SBI relative to LBI
3. Tutorial
4. Applications / future outlook / challenges
5. Deepskies Lab SBI projects

Fermilab

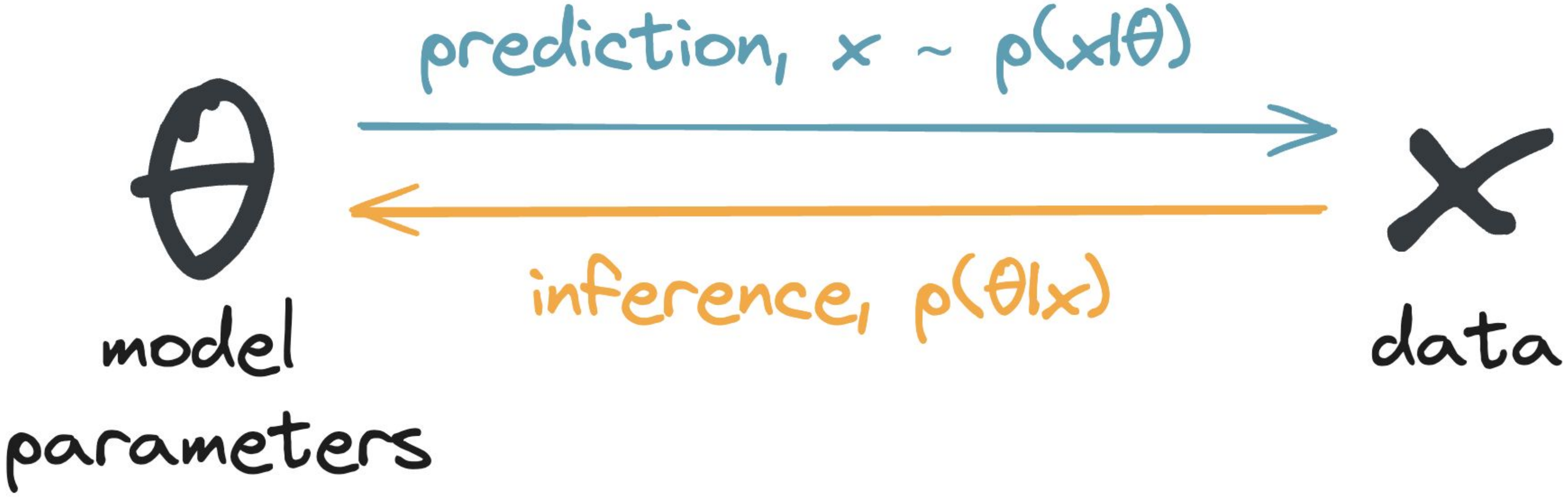
In physics, [and science in general], we love our simulations



These simulators are giving us a picture of the universe (x)



But what we really want is to be able to say thing statistically about the likely θs

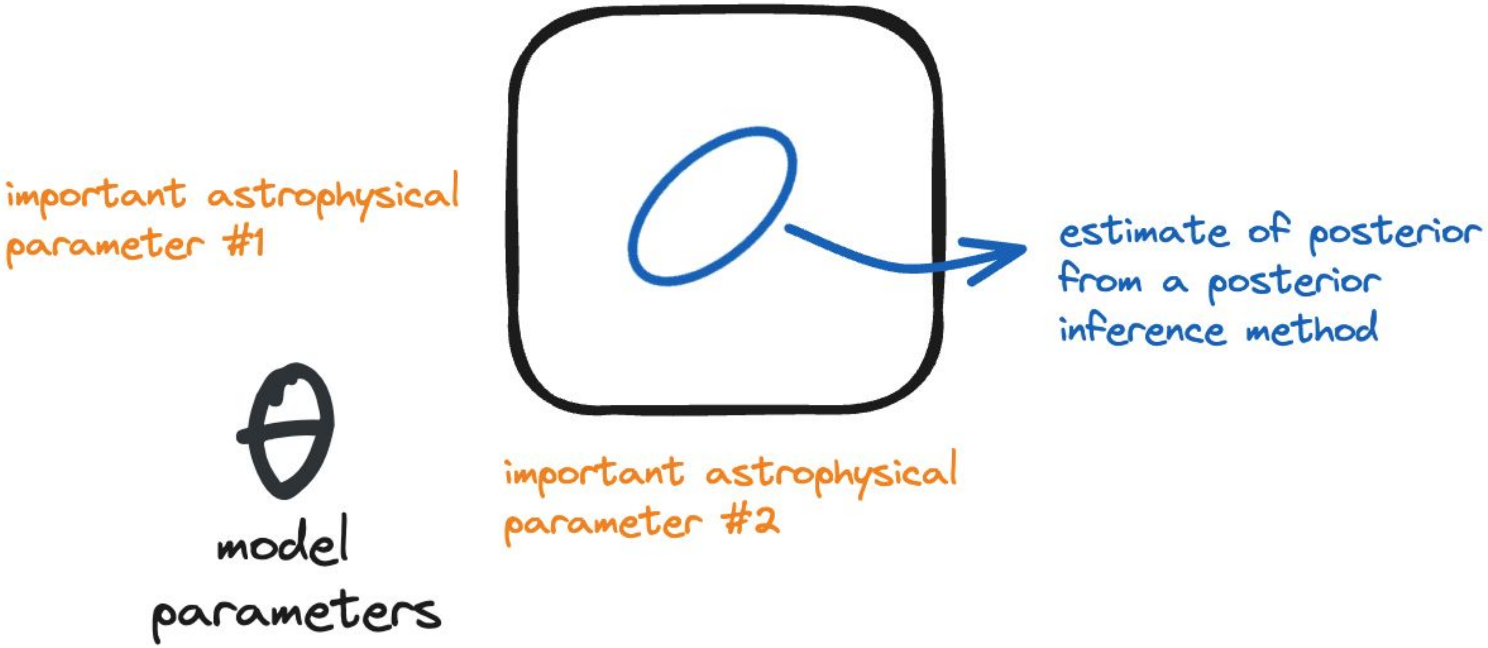


Inference

Given an observation x, we want to infer the *distribution* of probable parameters θ of the model

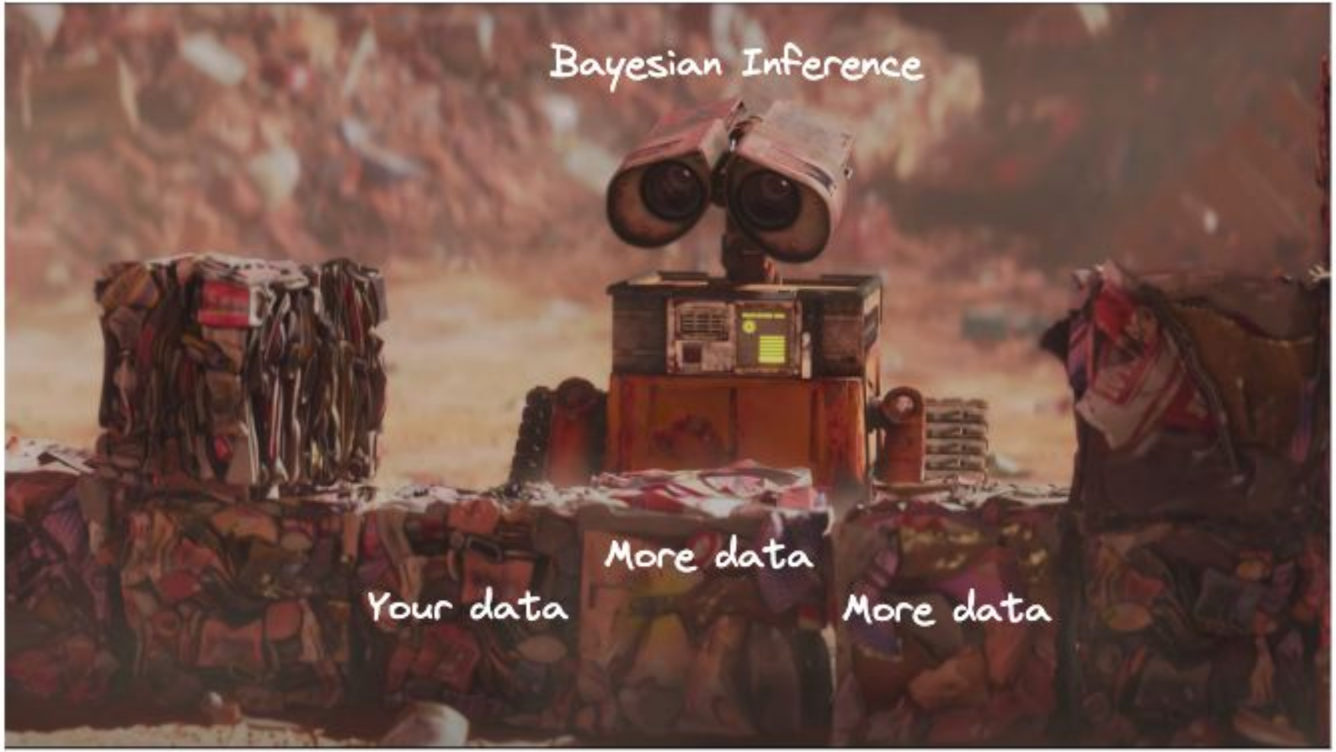
Inference

Given an observation x, we want to infer the *distribution* of probable parameters θ of the model, p(θ|x)



(Classic) Bayesian Inference (Bayes' Rule)

(Classic) Bayesian Inference (Bayes' Rule)



Bayesian Inference (Bayes' Rule), likelihood-based

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

Bayesian Inference (Bayes' Rule)

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

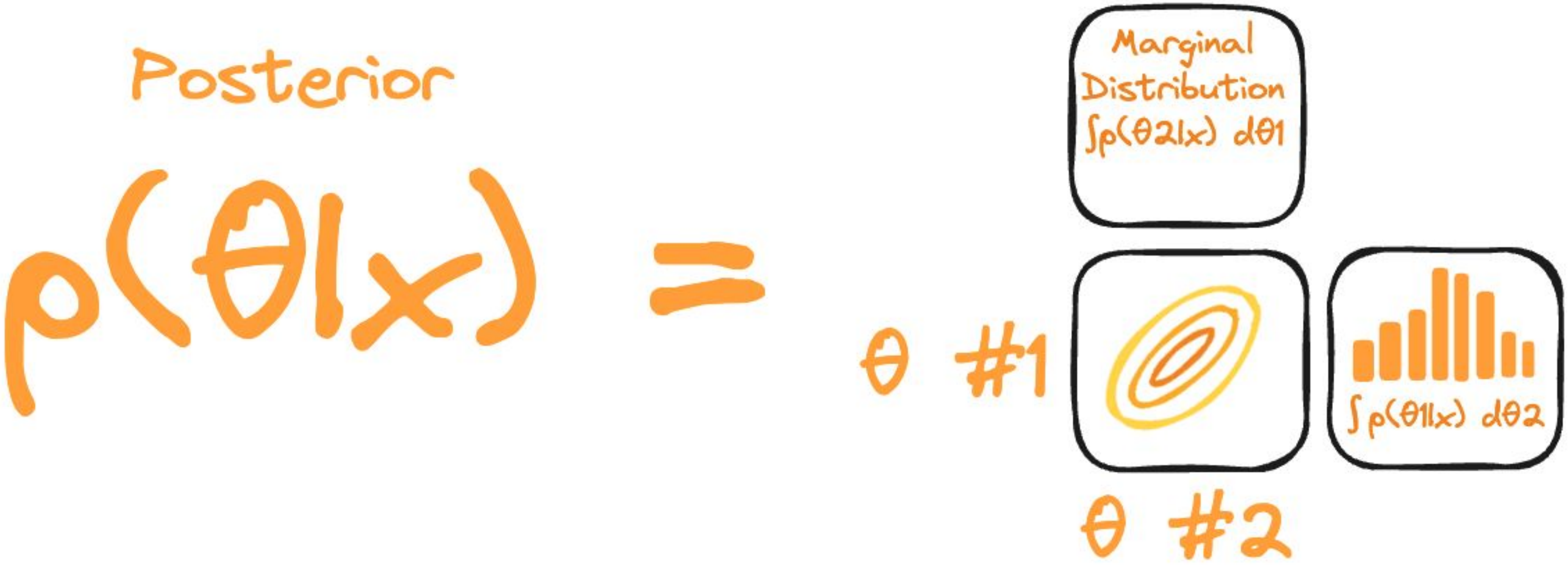
I'm going to try to convince you why this is often hard/impractical/intractable/impossible to do practically in physics

Bayesian Inference (Bayes' Rule)

Posterior

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

Bayesian Inference (Bayes' Rule)



Bayesian Inference (Bayes' Rule)

$$p(\theta|x) = \frac{\overbrace{p(x|\theta)}^{\text{Likelihood}}\, p(\theta)}{p(x)}$$

Bayesian Inference (Bayes' Rule)

Likelihood

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

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

Bayesian Inference (Bayes’ Rule)

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

Likelihood

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

Bayesian Inference (Bayes' Rule)

Prior

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

Bayesian Inference (Bayes' Rule)

$$p(\theta|x) = \frac{p(x|\theta)\,p(\theta)}{\underbrace{p(x)}_{\text{"Evidence"}}}$$

Bayesian Inference (Bayes' Rule)

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

"Evidence" $p(x)$

$$= \int p(x|\theta)\,p(\theta)\,d\theta$$

Fermilab

Likelihood-based inference versus **simulation**-based

Likelihood-based inference: Bayes' rule (analytically do some math), MCMC sampling approaches

Simulation-based inference: ????

Bayesian Inference (Bayes' Rule)

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

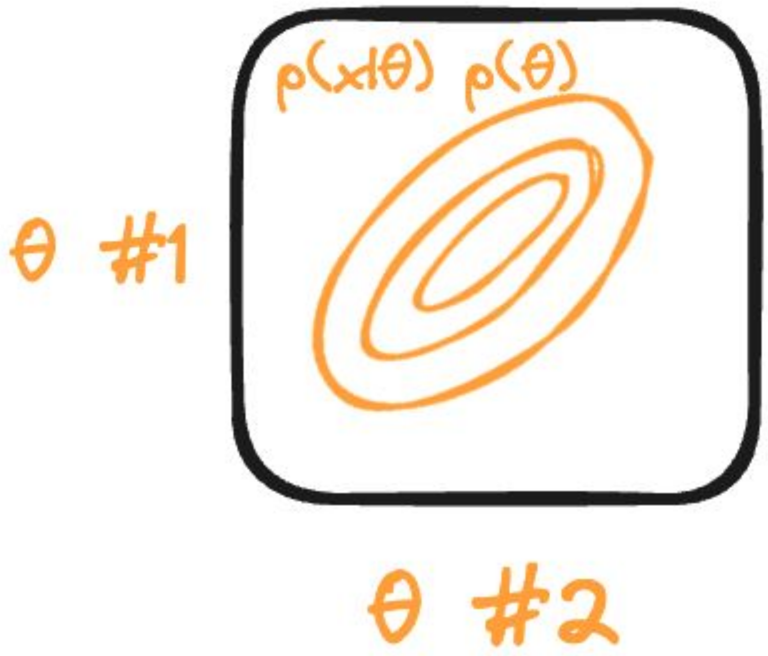
"Evidence" $p(x)$

$$= \int p(x|\theta)\,p(\theta)\,d\theta$$

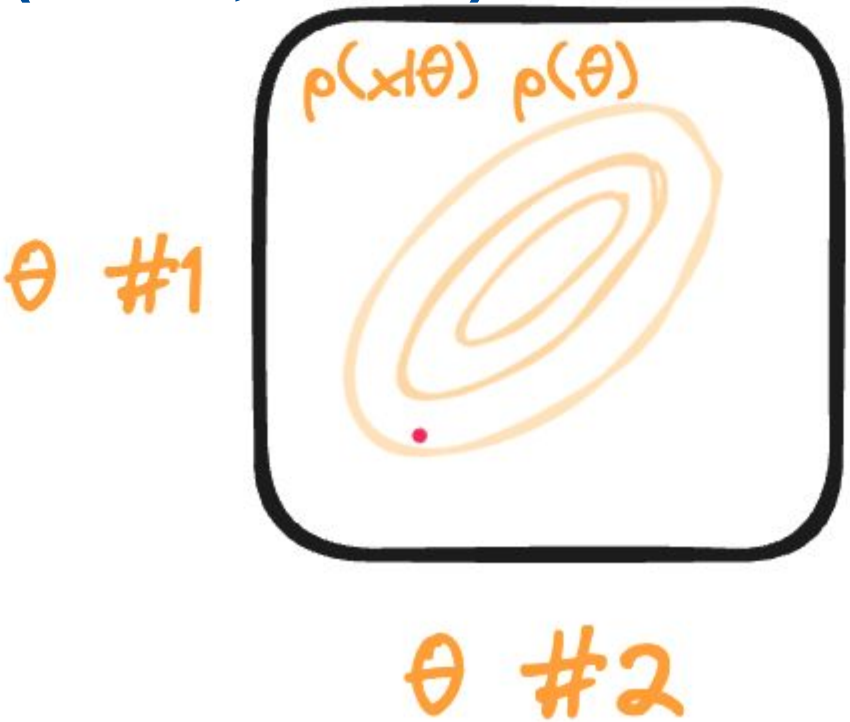
Likelihood-based inference

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

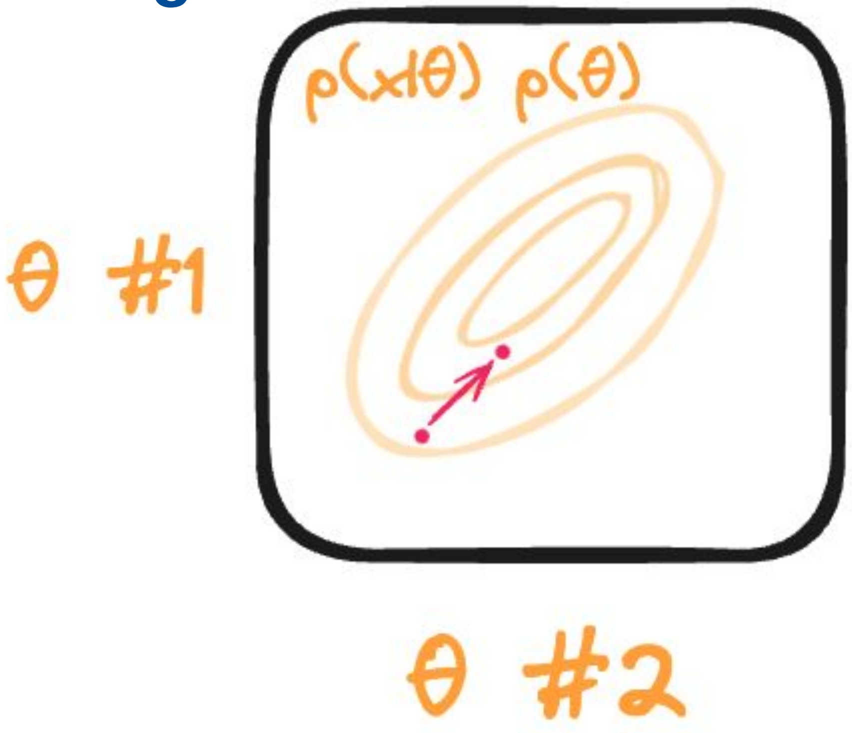
The goal of MCMC is to sample from the approximate posterior manifold



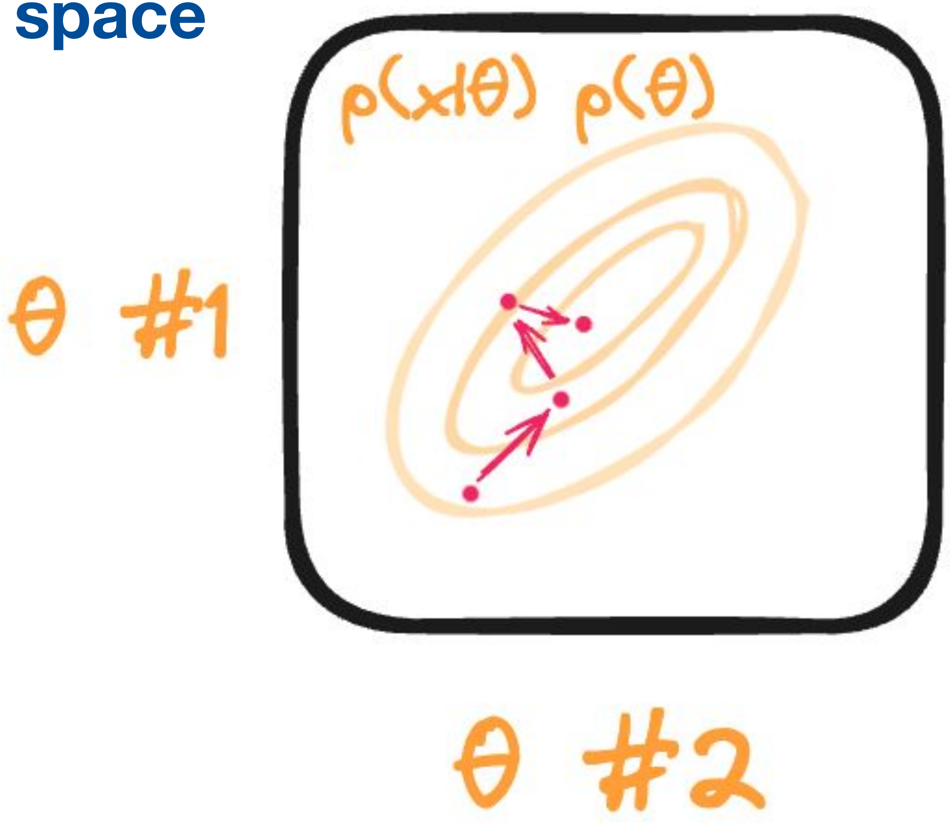
MCMC works iteratively by evaluating the likelihood and prior at a given (theta1, theta2) value



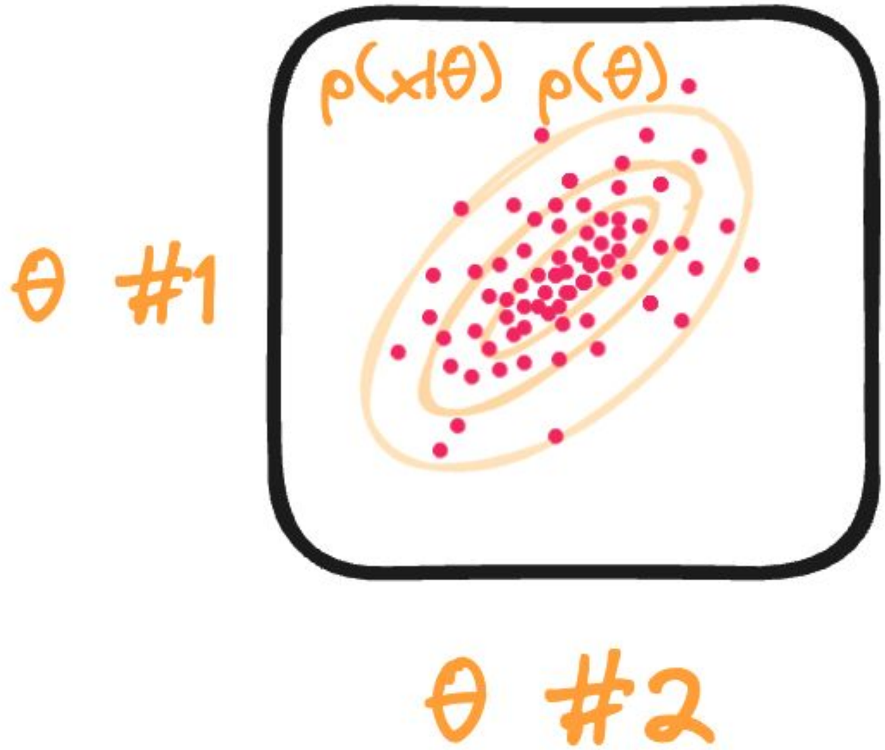
Then it takes a step; this can use a gradient or not depending on the algo



By taking a series of steps, it will walk towards areas of high probability space



The idea is to do this enough times with enough walkers that you build up a picture of the approximate posterior



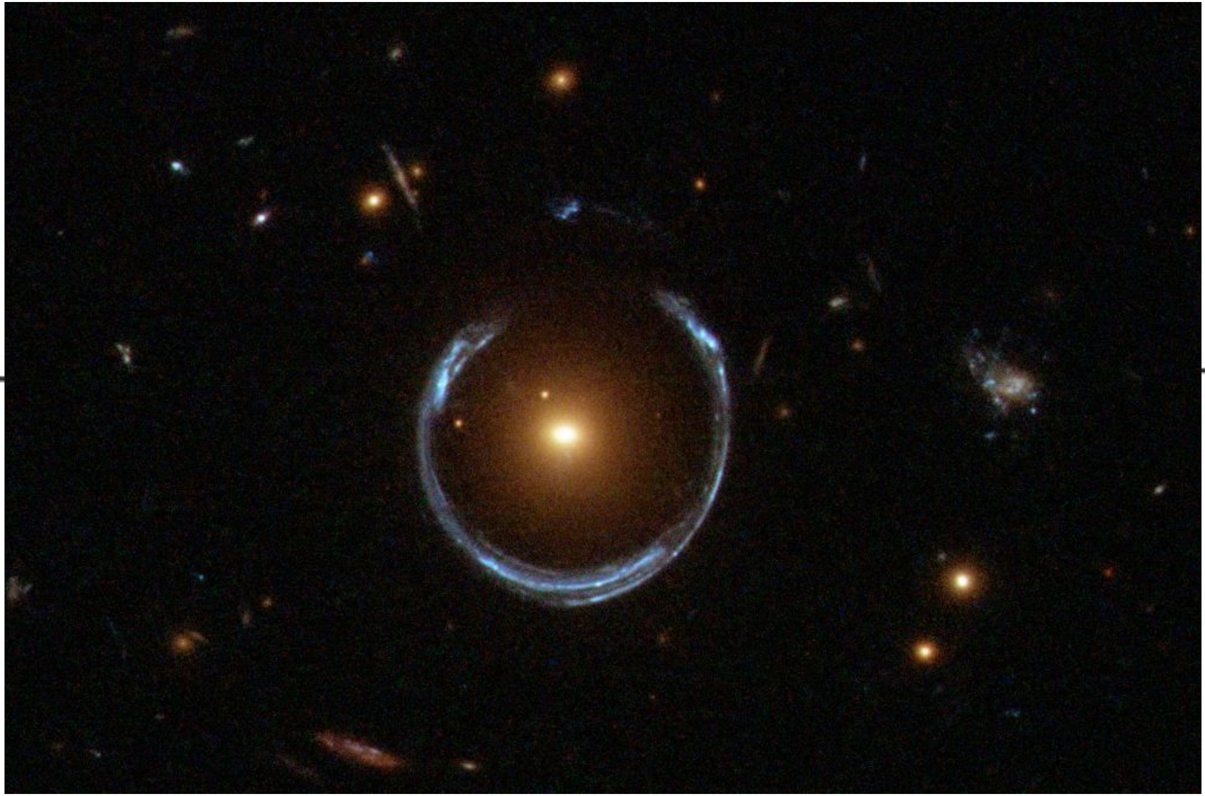
But back to the elephant in the room - what does calculating this likelihood entail?

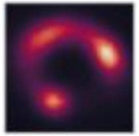
Likelihood

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

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

An example simulation from Deepskies lab; generating images of gravitational strong lenses





Parameters of interest

θ

Stuff like...
sersic profile
location
einstein radius
ellipticity, etc

An example simulation from Deepskies lab; generating images of gravitational strong lenses

[Morgan+2021](#)
Birrer+2021
Birrer & Amara 2018

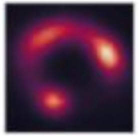
deepskies/
deeplenstronomy

A pipeline for versatile strong lens sample simulations

9 Contributors · 28 Issues · 6 Discussions · 25 Stars · 7 Forks

Parameters of interest

X $\leftarrow$ $\leftarrow$ θ



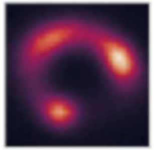
$$x \sim p(x|\theta)$$

Stuff like...
sersic profile
location
einstein radius
ellipticity, etc

Latent parameters

Observables

x



Cosmic evolution

cosmo ← θ

Parameters of interest

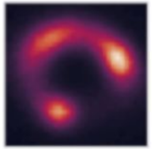
cosmological parameters, coupling of dark matter and galaxies

Stuff like... sersic profile location einstein radius ellipticity, etc

Latent parameters

Observables

x



Ray tracing and thin lens approximation

light ← cosmo ← θ

deterministic physics equations, integrating along the line of sight

Cosmic evolution

cosmological parameters, coupling of dark matter and galaxies

Parameters of interest

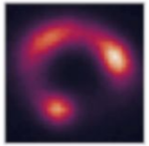
Stuff like... sersic profile location einstein radius ellipticity, etc

Fermilab

Latent parameters

Observables | Detector / survey interactions | Ray tracing and thin lens approximation | Cosmic evolution | Parameters of interest

$x \leftarrow$ detect $\leftarrow$ light $\leftarrow$ cosmo $\leftarrow \theta$



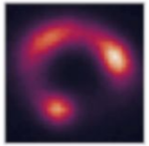
Poisson noise, detector properties, various stochastic processes

deterministic physics equations, integrating along the line of sight

cosmological parameters, coupling of dark matter and galaxies

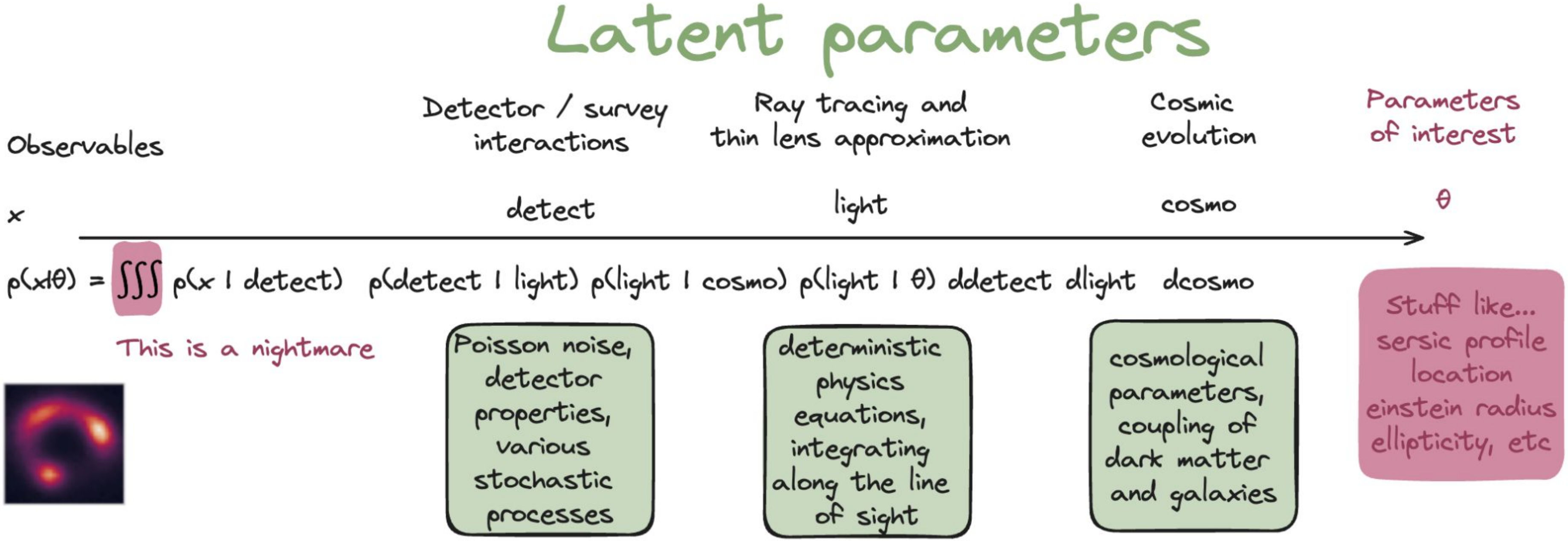
Stuff like... sersic profile location einstein radius ellipticity, etc

$$x \sim p(x|\theta)$$

Observables	Detector / survey interactions	Ray tracing and thin lens approximation	Cosmic evolution	Parameters of interest
x	detect	light	cosmo	θ
	Poisson noise, detector properties, various stochastic processes	deterministic physics equations, integrating along the line of sight	cosmological parameters, coupling of dark matter and galaxies	Stuff like... sersic profile location einstein radius ellipticity, etc

$x \leftarrow \text{detect} \leftarrow \text{light} \leftarrow \text{cosmo} \leftarrow \theta$

Likelihood-based inference is challenging if the simulation is very high dimensional with lots of params



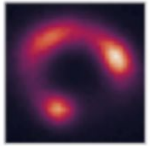
Likelihood-based inference is challenging if the simulation is very high dimensional with lots of params

Latent parameters

Observables	Detector / survey interactions	Ray tracing and thin lens approximation	Cosmic evolution	Parameters of interest
x	detect	light	cosmo	θ

$$p(x|\theta) = \iiint p(x \mid \text{detect})\; p(\text{detect} \mid \text{light})\; p(\text{light} \mid \text{cosmo})\; p(\text{light} \mid \theta)\; d\text{detect}\; d\text{light}\; d\text{cosmo}$$

This is a nightmare



detect	light	cosmo	θ
Poisson noise, detector properties, various stochastic processes	deterministic physics equations, integrating along the line of sight	cosmological parameters, coupling of dark matter and galaxies	Stuff like... sersic profile location einstein radius ellipticity, etc

Idea credit: Kyle Cranmer

Fermilab

Disadvantages of MCMC / Likelihood-based inference

1. You need to evaluate the likelihood
2. You have to do this for every piece of data

Disadvantages of MCMC / Likelihood-based inference

1. You need to evaluate the likelihood
2. You have to do this for every piece of data (trash)

But what if I told you:
Is not always necessary to evaluate the likelihood

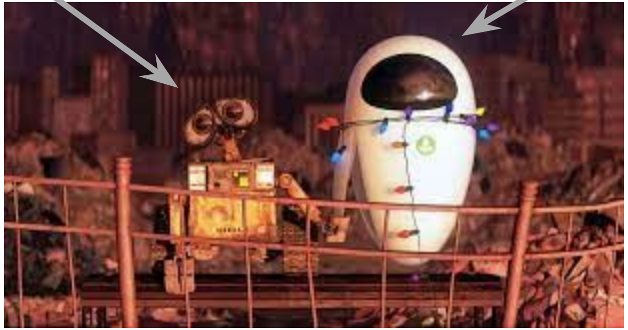
Simulation-based inference does not require that you write out or evaluate a likelihood

A word on "likelihood-free" inference

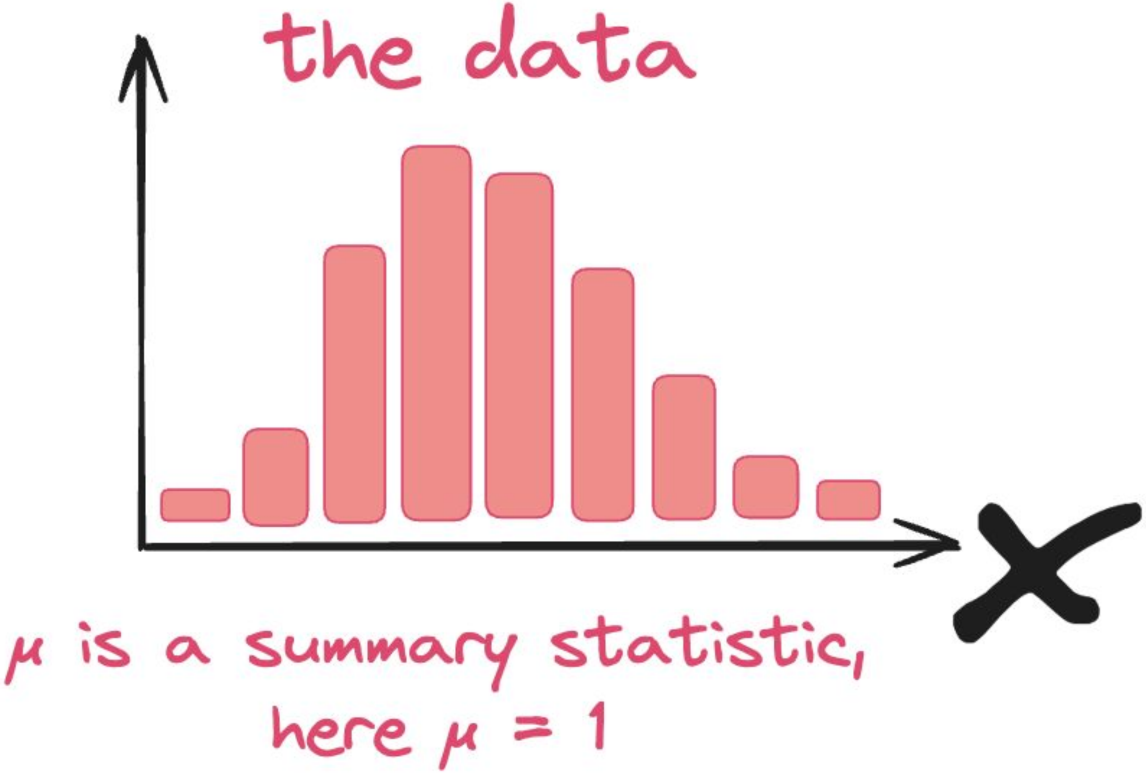
- Can still be floating around in there, "implicit likelihood"
- There might still be a likelihood you can write out

Approximate bayesian computation (ABC) is an example of SBI

Marriage of Bayesian inference and computation

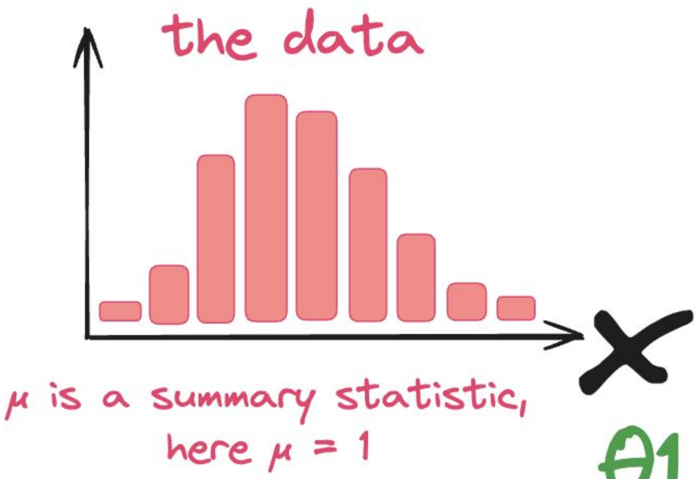


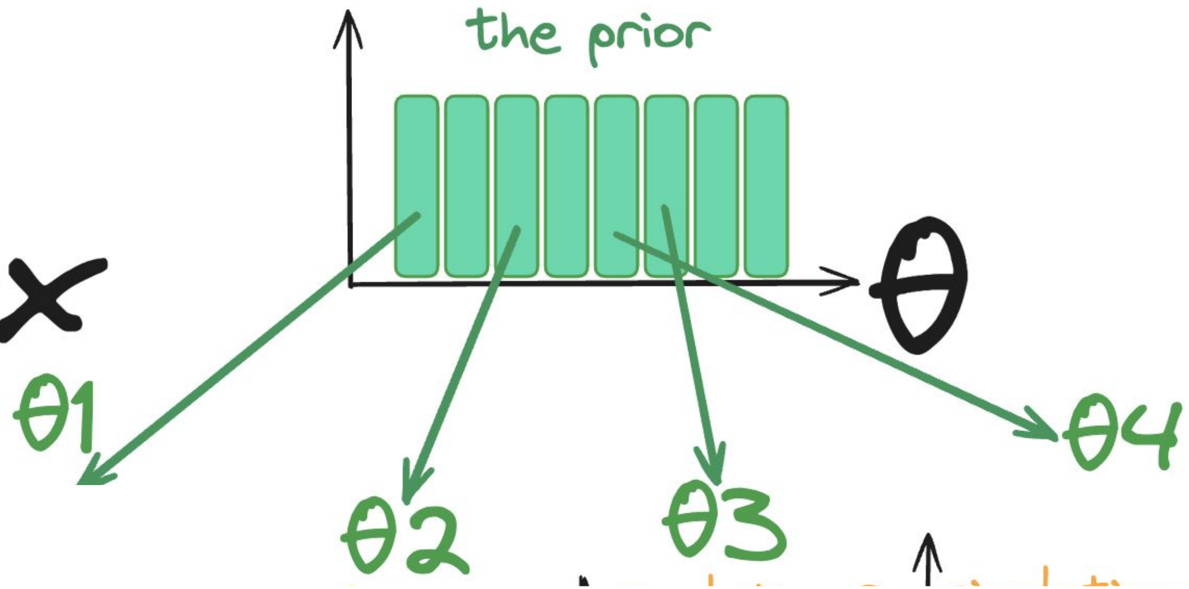
If you have access to an efficient simulation…


the data
x
μ is a summary statistic,
here μ = 1

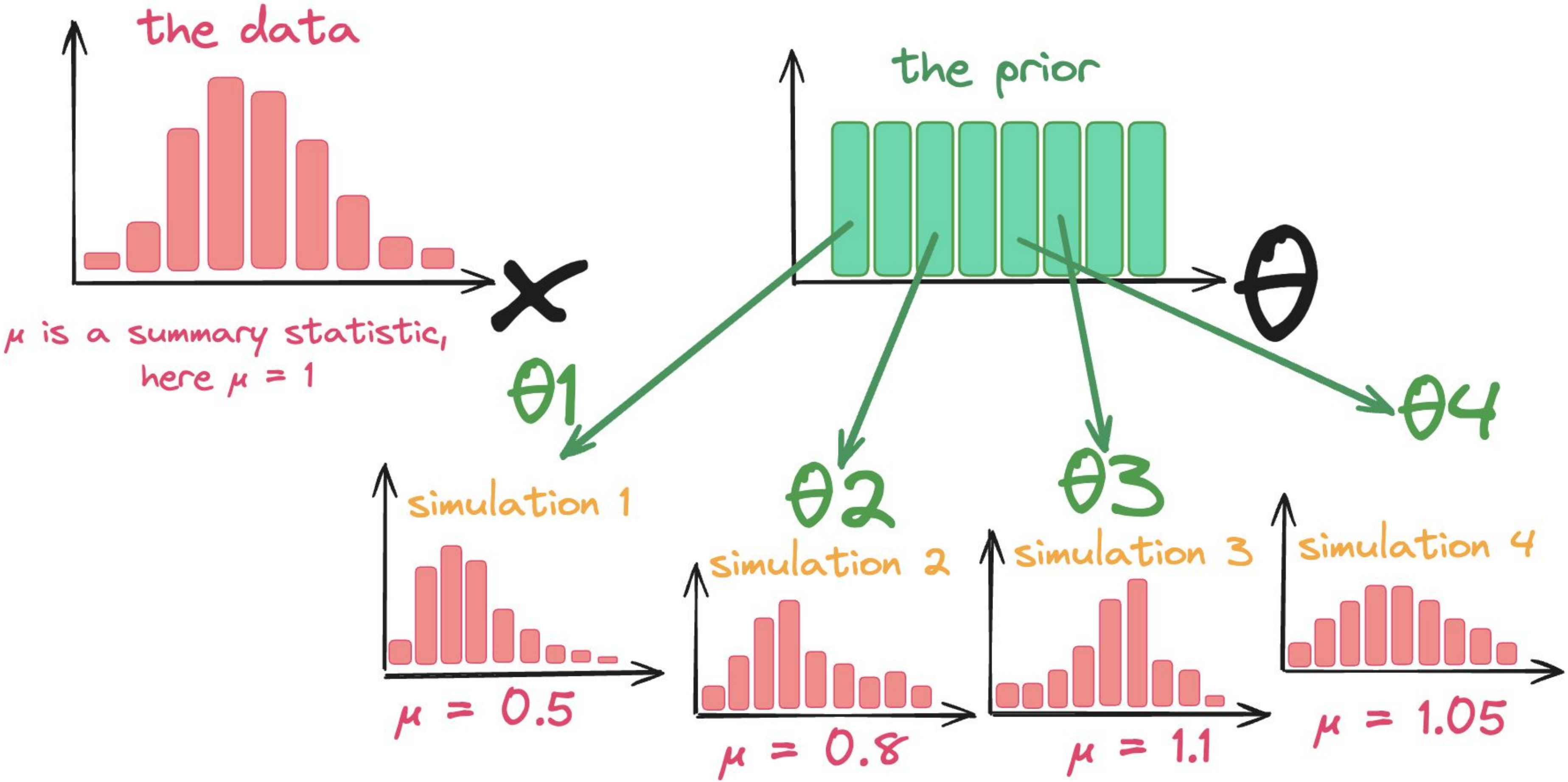

the data
x
μ is a summary statistic,
here μ = 1
θ
the sim
x


the data
θ
the sim
x
x
μ is a summary statistic,
here μ = 1
the goal
θ

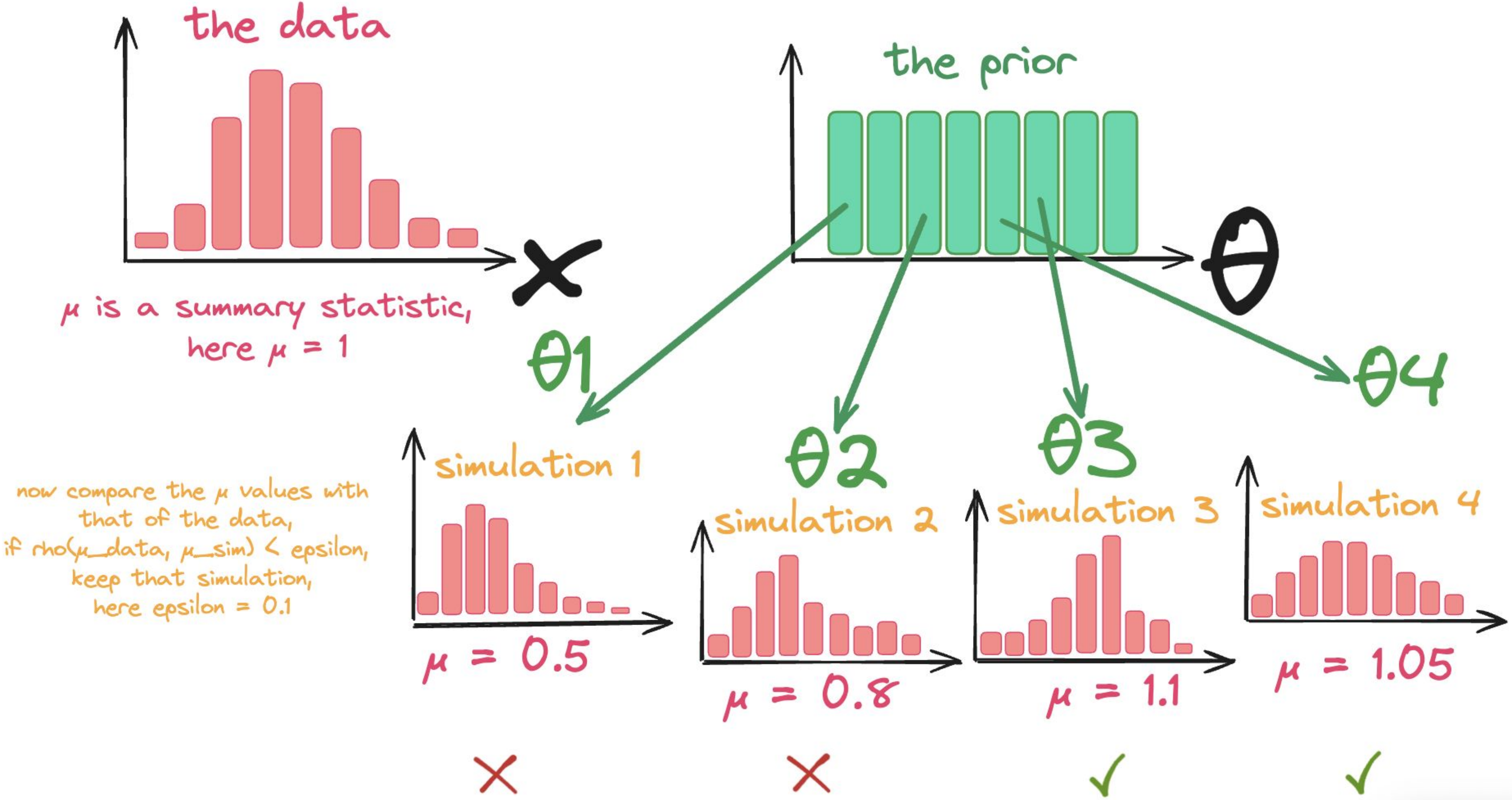




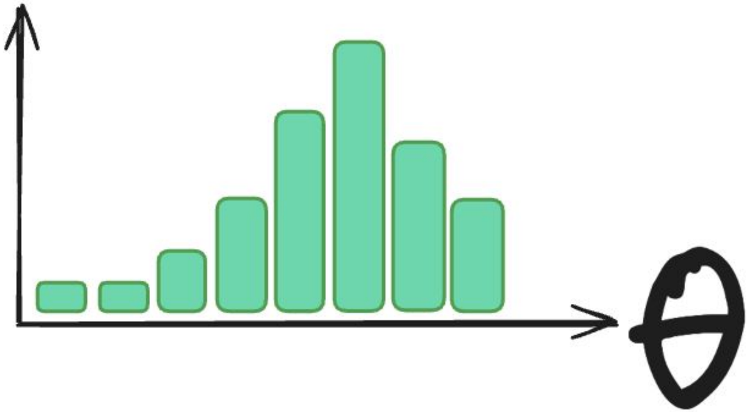
[Miles Cranmer Blog]()



[Miles Cranmer Blog]

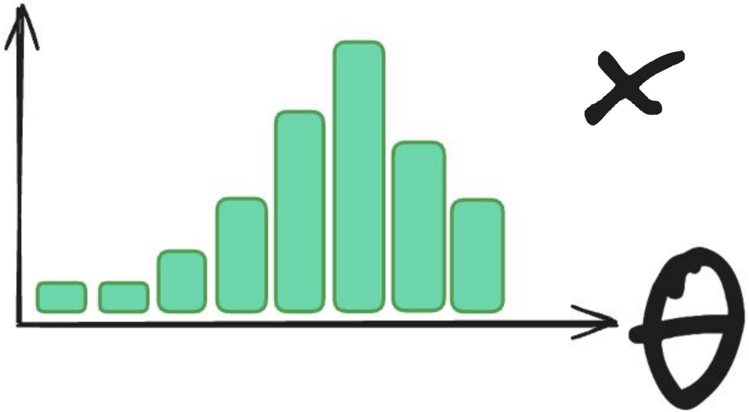

the data
X
μ is a summary statistic,
here μ = 1
the prior
θ
θ1
θ2
θ3
θ4
now compare the μ values with
that of the data,
if rho(μ_data, μ_sim) < epsilon,
keep that simulation,
here epsilon = 0.1
simulation 1
μ = 0.5
simulation 2
μ = 0.8
simulation 3
μ = 1.1
simulation 4
μ = 1.05

combine the θ values from the simulators that pass the test
use these to construct the posterior distribution as a fxn of θ



ABC involves the likelihood implicitly via sampling

combine the θ values from the simulators that pass the test
use these to construct the posterior distribution as a fxn of θ



$x \sim p(x|\theta)$

combine the θ values from the simulators that pass the test

use … of θ

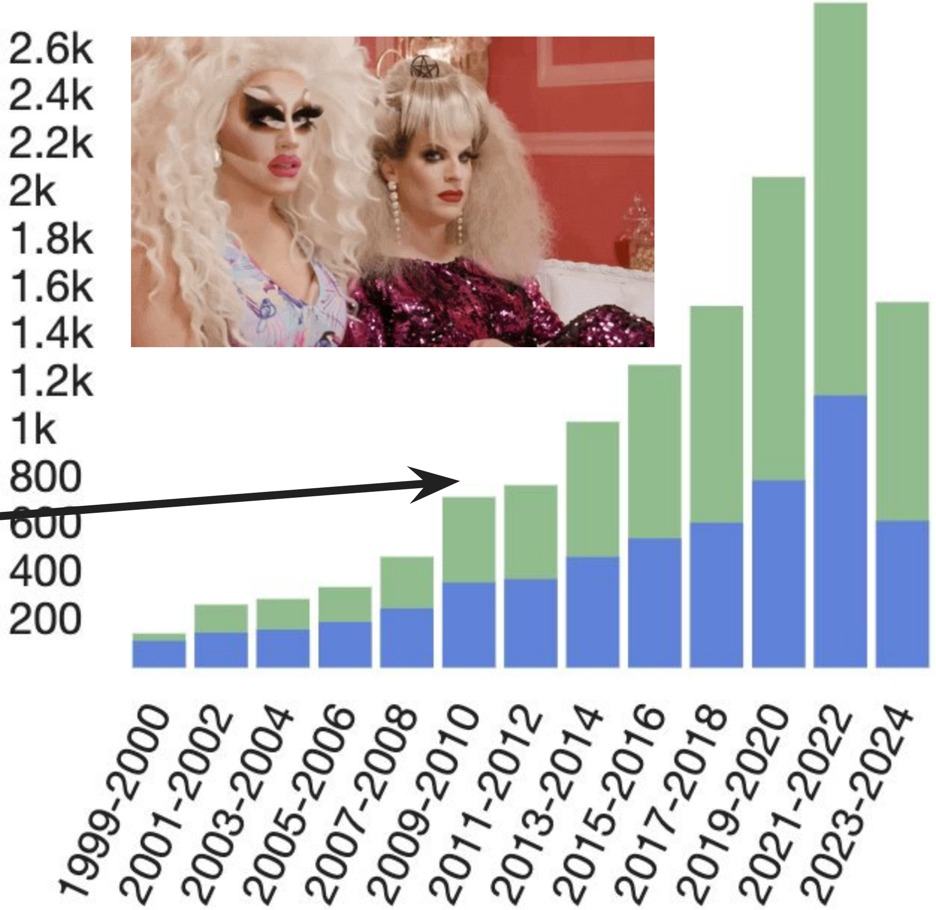
Surprise! ABC is an early form of simulation-based inference! It uses a simulation and not an explicit likelihood to approximate the posterior landscape.

Simulation-based inference has undergone a recent glow-up

First normalizing flows article I could find:

Agnelli+ 2010 “Clustering and Classification through Normalizing Flows in Feature Space”

Rezende & Mohamad 2015



The SBI landscape circa 2020

Credit: [Cranmer et al (2020)]
"The Frontier of SBI"

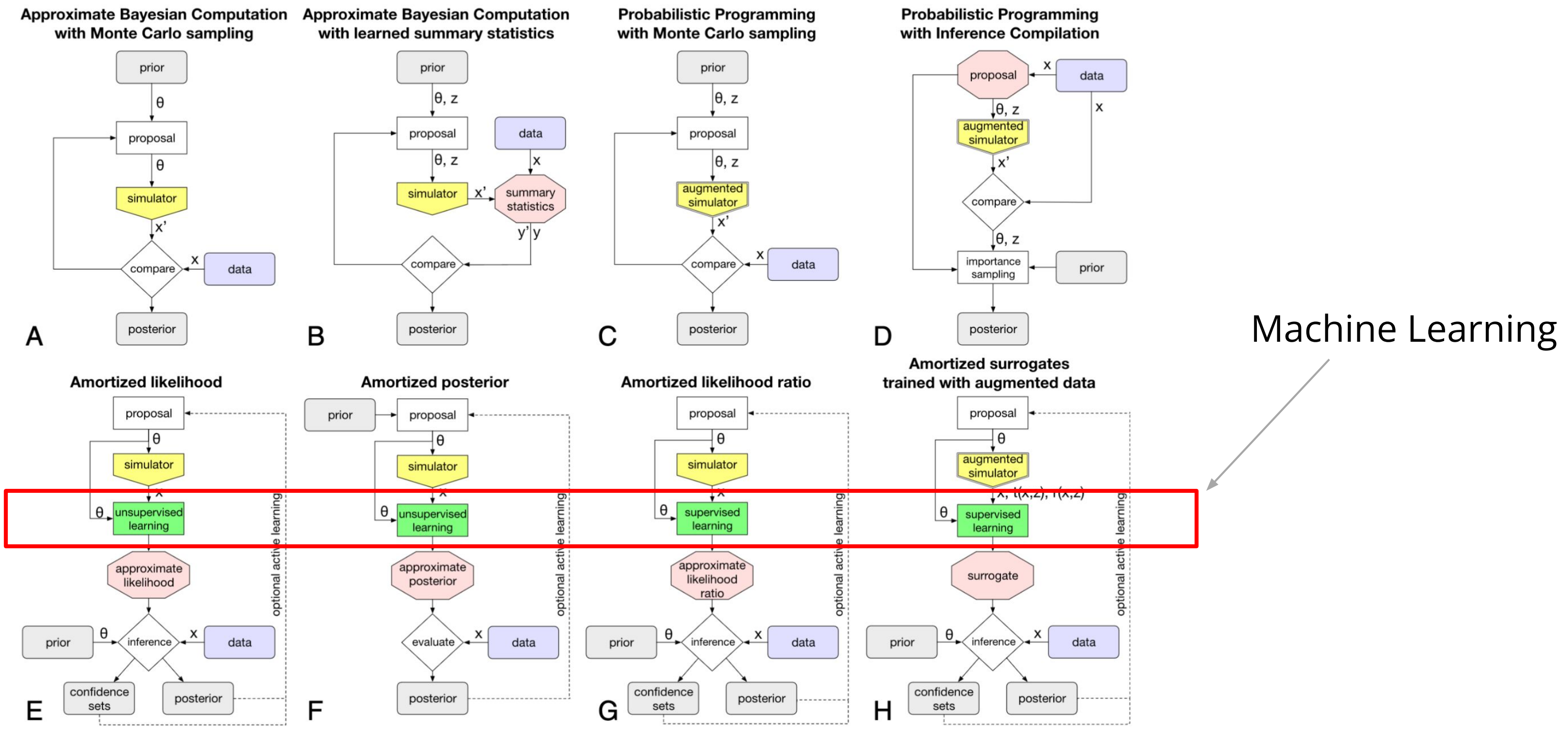
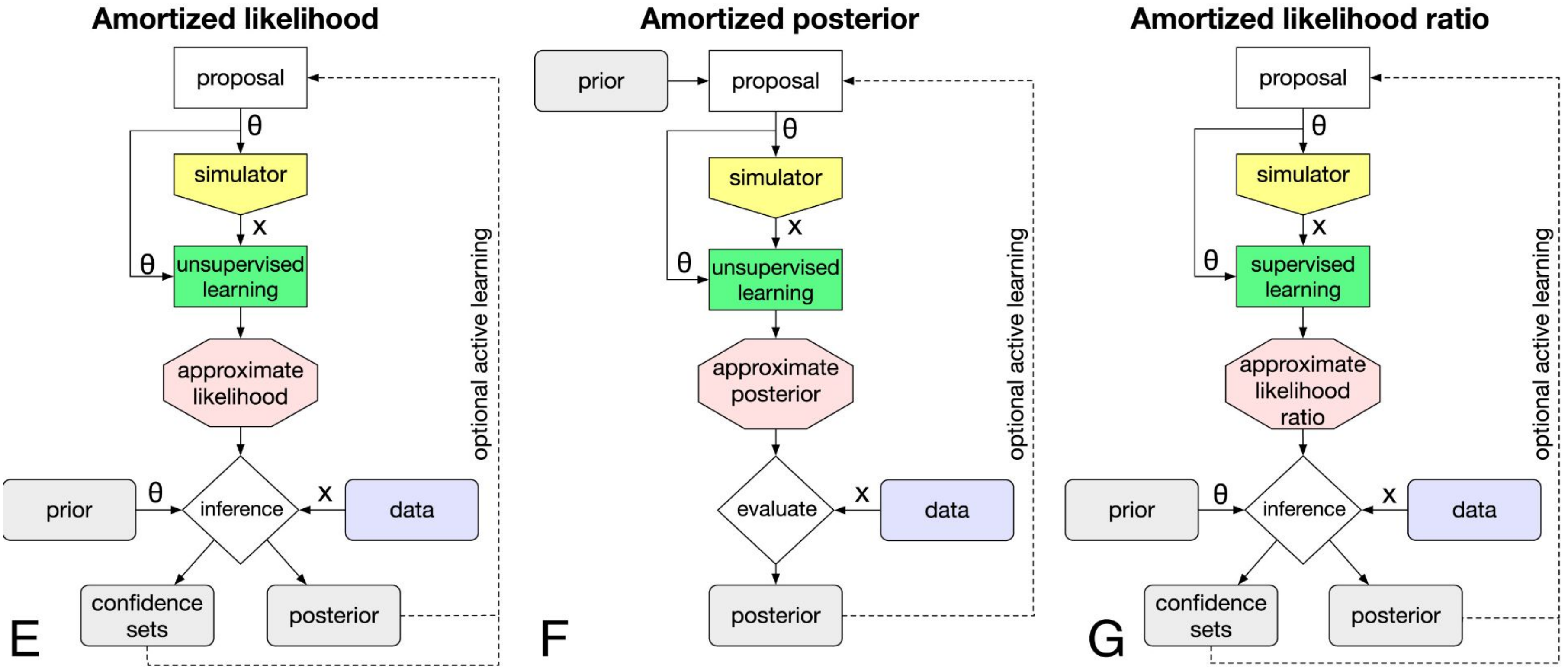
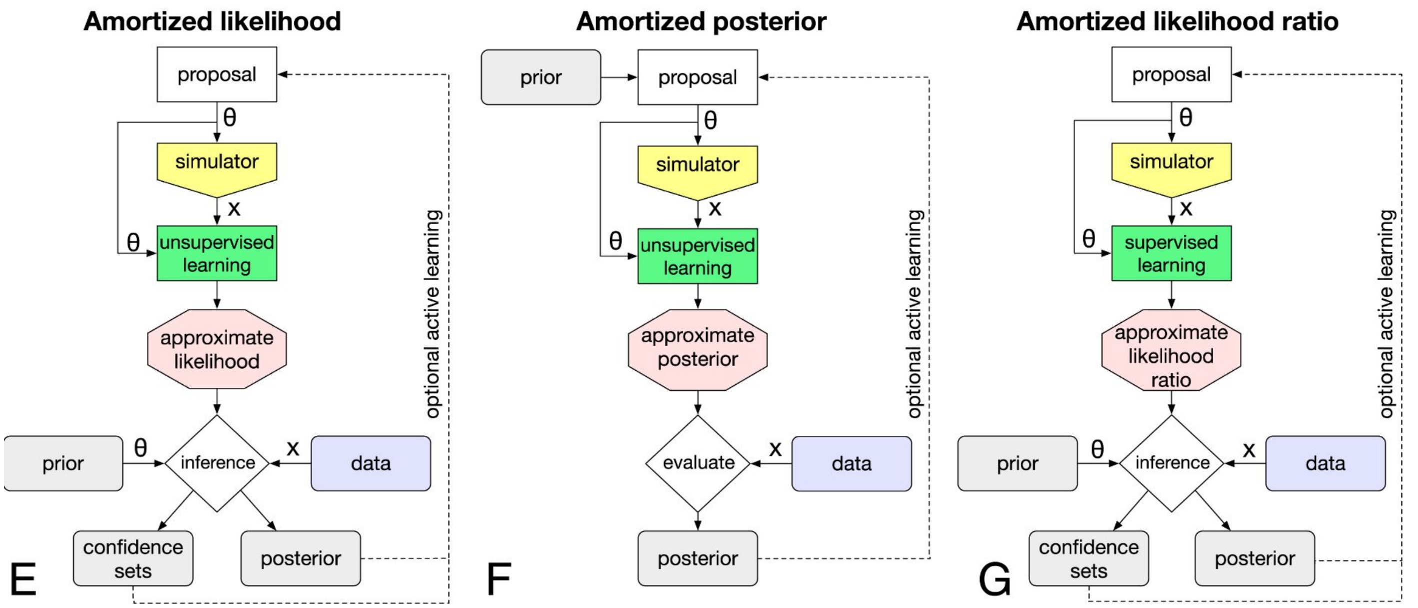


Fig. 1. (A–H) Overview of different approaches to simulation-based inference.

A HUGE advantage of modern SBI methods is that they are amortized



A HUGE advantage of modern SBI methods is that they are amortized



Likelihood Posterior Likelihood ratio

Targets of SBI

- Likelihood ratio
- Likelihood / evidence = posterior / prior
- Likelihood (Neural likelihood estimation), estimating the probability density of the data conditional on parameters
- Neural posterior estimation
- Dimensionality of data and parameter vector

Advantages of SBI

- No likelihood required, skips integrating over latent parameters
- You don't have to re-run the inference for every data point
- Very advantageous in cases of large latent dimensionality

Fermilab

Additional resources

- Talk by Stephen Green about gravitational wave parameter estimation
- Talk by Kyle Cranmer also about SBI
- https://www.youtube.com/watch?v=315xKcYX-1w&ab_channel=TheTWIMLAIPodcastwithSamCharrington → an ML podcast with George Papamakarios about masked autoregressive flows
- https://www.youtube.com/watch?v=7q4ueFiJjAY&ab_channel=KapilSachdeva → MADE paper review
- This awesome blog about flow models: https://lilianweng.github.io/posts/2018-10-13-flow-models/

Details of the neural density estimation

- Review of normalizing flows

- Masked autoregressive flows

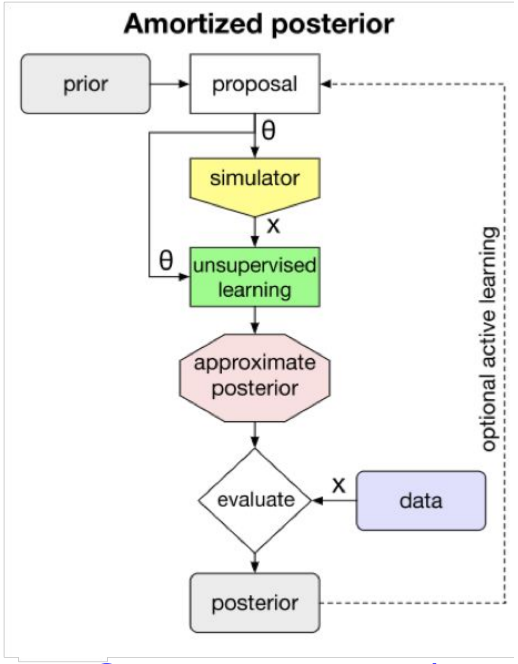
Fermilab

Modern SBI methods, neural posterior estimation (NPE)

Modern SBI methods leverage **machine learning** techniques to overcome the issues faced by traditional SBI methods.

The general workflow:

1. Pay upfront simulation cost of generating many pairs of $\{\theta_i|x_i\}$.
2. Use ML model (neural density estimators) to model the posterior distribution from the simulation data.
3. Evaluate new observations x_{new} using the trained model.



Credit: [Cranmer et al (2020)]

Sequential slide

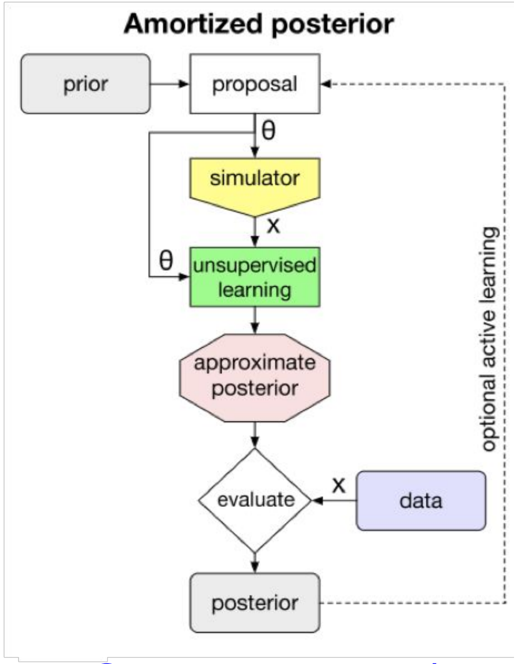
- When the posterior is way more concentrated than the prior you don’t really need to explore the likelihood everywhere in parameter space
- Motivates active learning aka sequential methods such as:
  - Sequential Neural LIkelihood Estimation [SNLE]
  - Sequential Neural Posterior Estimation [SNPE]
  - Sequential Neural Ratio Estimation [SNRE]
- NOT amortized, partially amortized

Modern SBI methods

Modern SBI methods leverage **machine learning** techniques to overcome the issues faced by traditional SBI methods.

The general workflow:

1. Pay upfront simulation cost of generating many pairs of $\{\theta_i | x_i\}$.
2. Use ML model (neural density estimators) to model the posterior distribution from the simulation data.
3. Evaluate new observations x_{new} using the trained model.

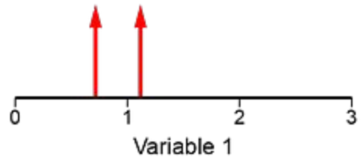


Credit: [Cranmer et al (2020)]

Neural density estimators

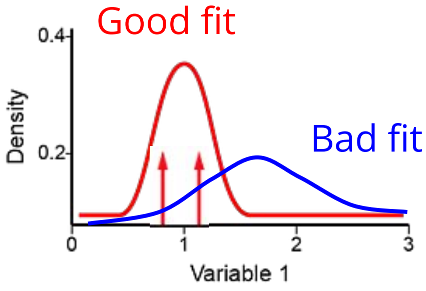
Problem statement: We have some data $\{x_1,\ldots x_n\}$ generated from a process $p(x)$. We want to fit a parametric model, $q_\phi(x)$ to approximate $p(x)$.

- We want to fit $q_\phi(x)$ to the data such that $q_\phi(x)$ is **most likely** to generate $\{x_1,\ldots x_n\}$.



Training samples
Each arrow represent a sample

Generative model learning algorithm →



Learned Distribution

$$L(\boldsymbol{\phi}) = \frac{1}{N}\sum_n \log q_{\boldsymbol{\phi}}(\mathbf{x}_n)$$

Credit: Jason Poh

Neural density estimators

There are non-neural density estimators, think Gaussians (parametric) or kernel density estimators (non-parametric)

But **neural** network based density estimators have more freedom

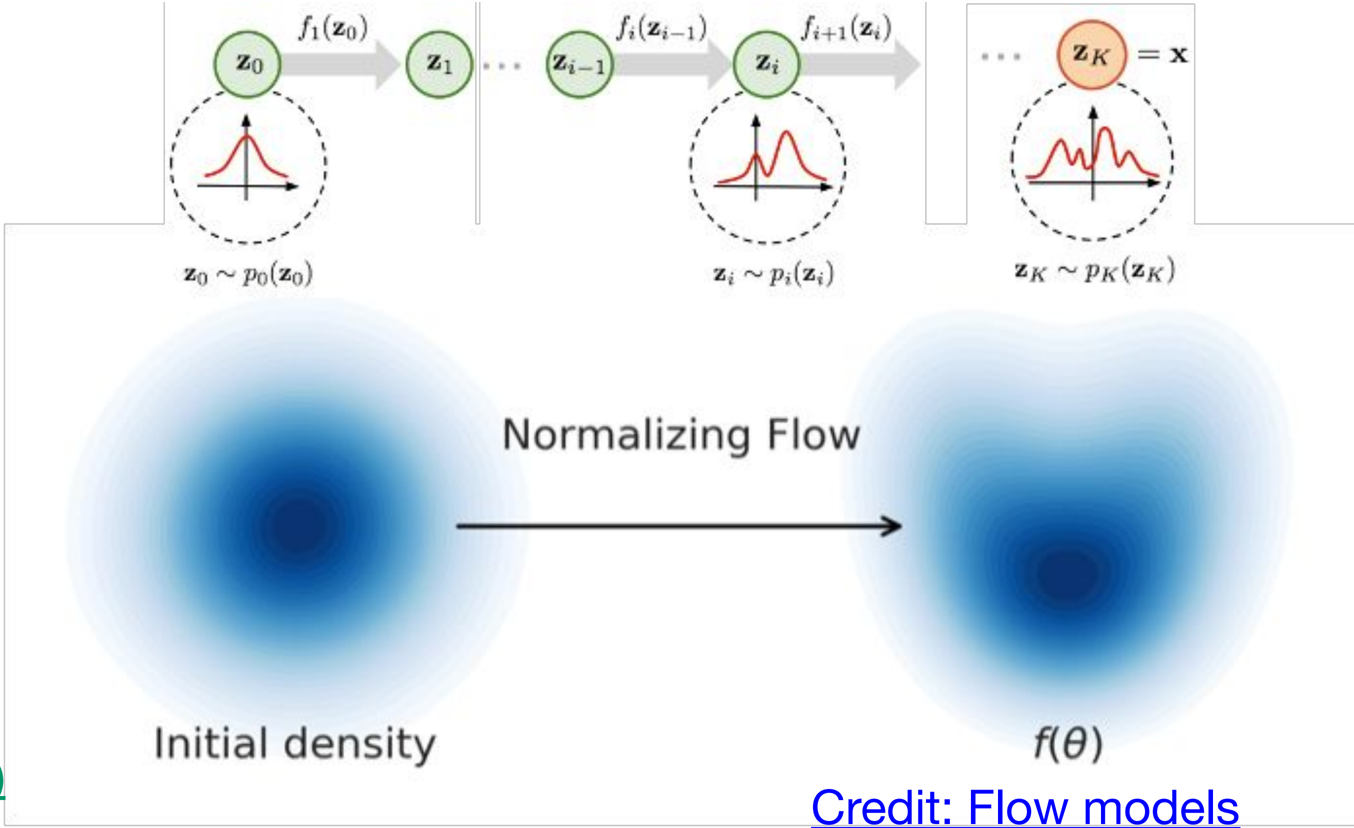
5c68f1206823815f66102863-Paper.pdf

Fast ϵ-free Inference of Simulation Models with Bayesian Conditional Density Estimation

George Papamakarios
School of Informatics
University of Edinburgh
g.papamakarios@ed.ac.uk

Iain Murray
School of Informatics
University of Edinburgh
i.murray@ed.ac.uk

Building blocks of modern NDEs

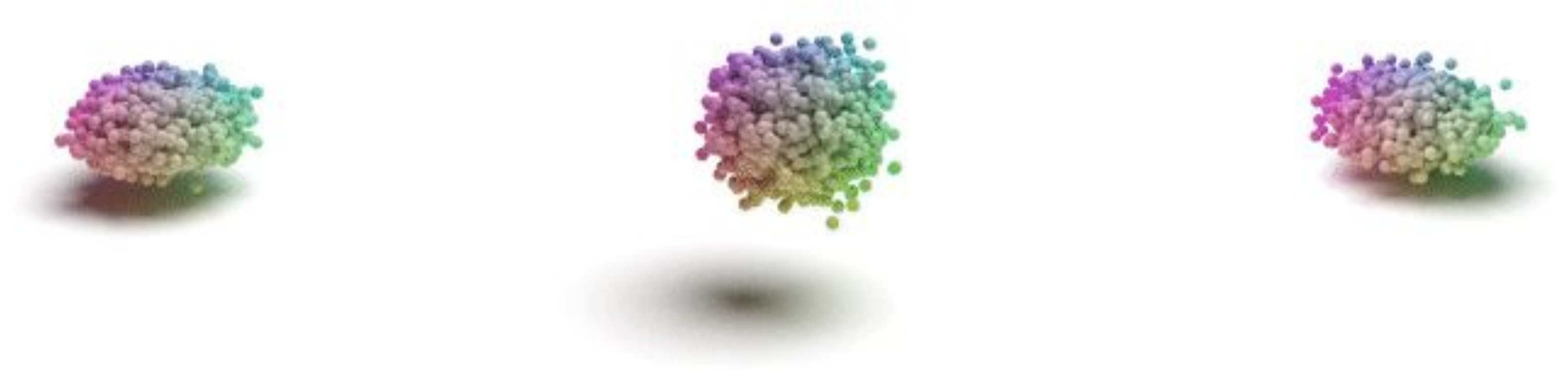


[A 1 minute description]

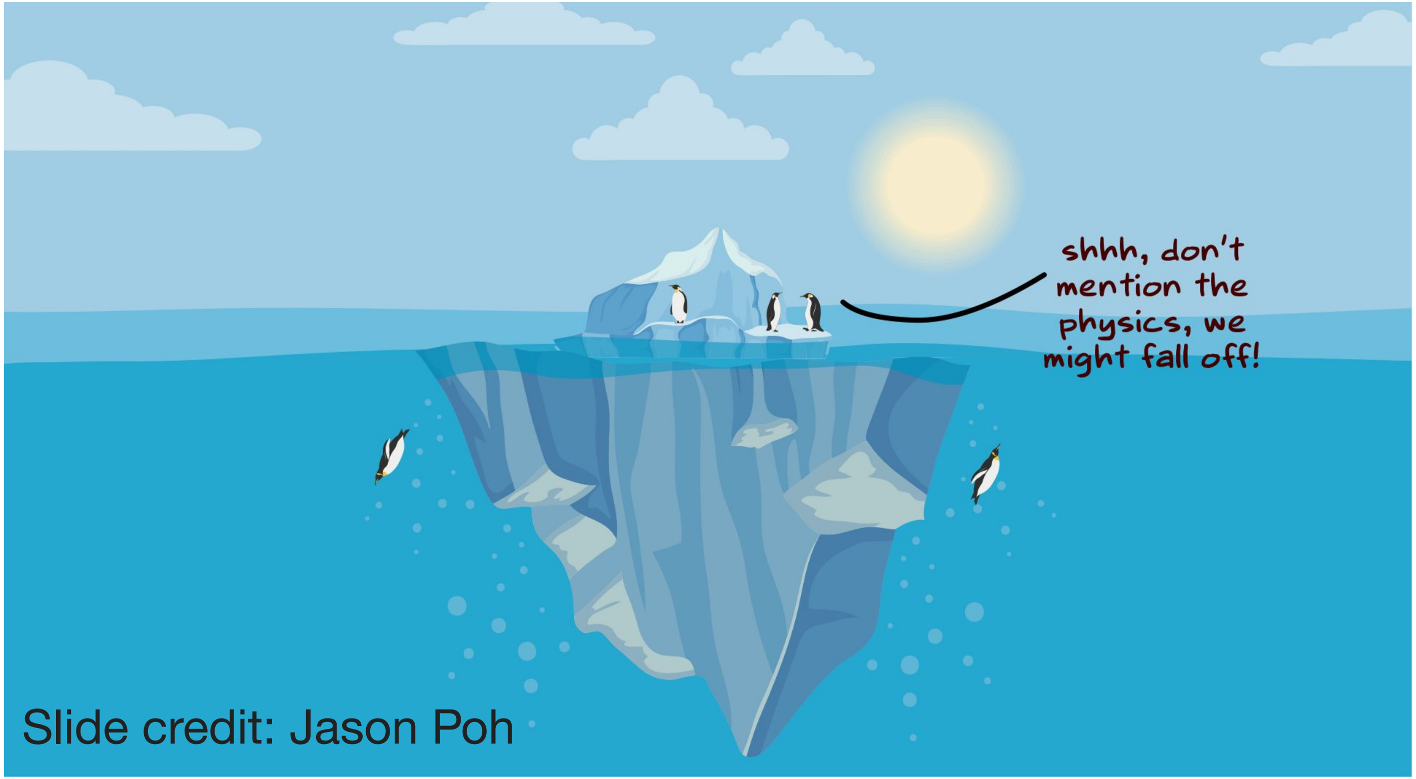
[Video Tutorial (12 mins)]

[Credit: Flow models]

Normalizing flows (and the other methods within this family) can transform any distribution into a simpler distribution in an invertile way (bijective)



Masked Autoregressive Flows



Autoregressive models model densities conditionally

- A class of models where a target joint distribution is factorized over n-dimensional probability conditionals, and those conditionals are modeled in turn.
- For example, if we are interested in the probability density of an image in an autoregressive model, the joint probability of an image could be the combination of the probability of all its pixels [(as was used in PixelCNN)]

$$p(\boldsymbol{x}) = p(x_1, x_2, \dots, x_n)$$

$$p(\boldsymbol{x}) = p(x_1)p(x_2) \dots p(x_n)$$

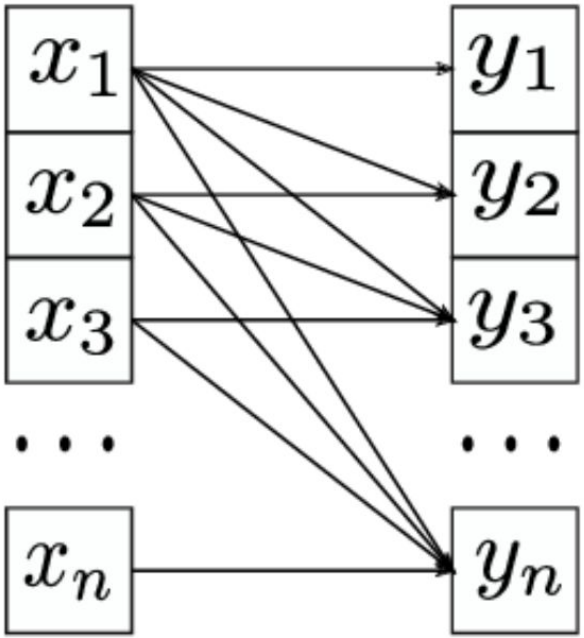
$$p(\boldsymbol{x}) = \prod_{i=1}^{n} p(x_i)$$

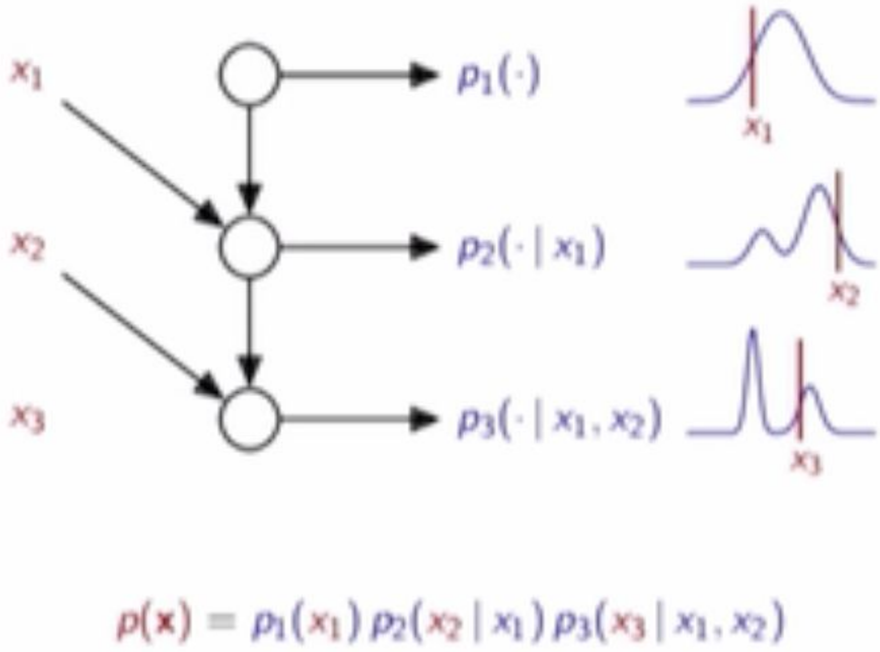
$$\underbrace{p(x)}_{\text{Likelihood of image } \boldsymbol{x}} = \prod_{i=1}^{n} \underbrace{p(x_i | x_1, \dots, x_{i-1})}_{\text{Probability of i'th pixel value given all previous pixels}}$$

Credits: PixelCNN

Slide credit: Jason Poh

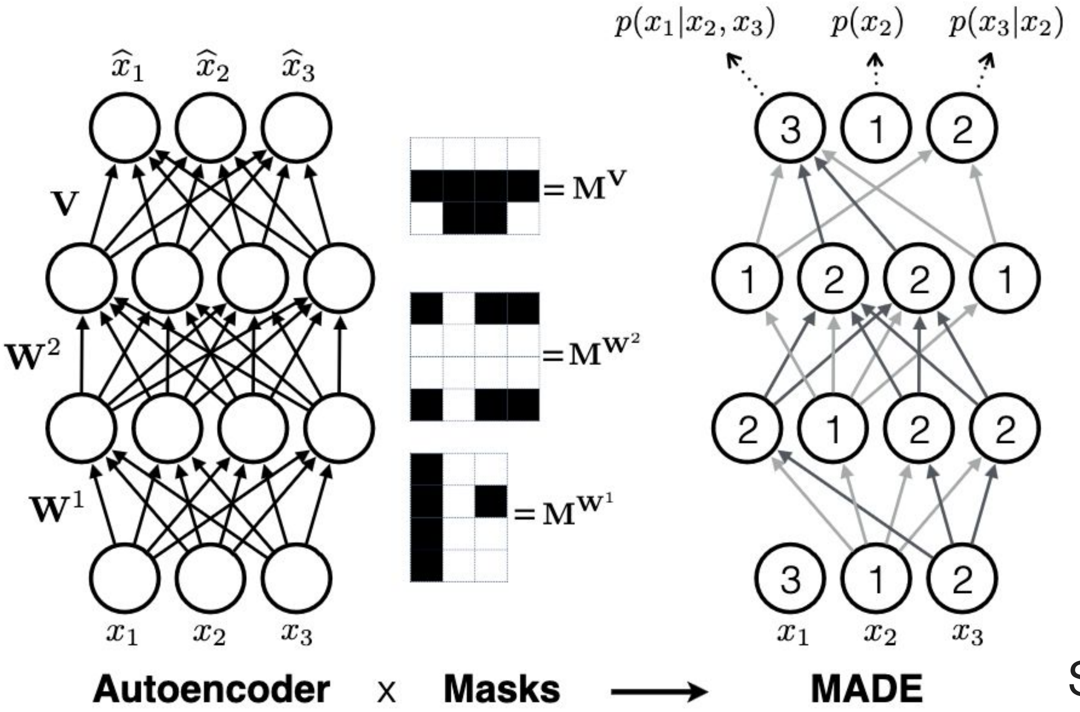
An autoregressive network uses the previous inputs to estimate the density





Masked Autoencoders for Density Estimation (MADE)

$$\widehat{x}_d = p(x_d \,|\, \mathbf{x}_{<d})$$



[Video Tutorial](#)
[Germain+2015](#) MADE paper
[Papamakarios+2018](#) MAF paper

Slide credit: Jason Poh

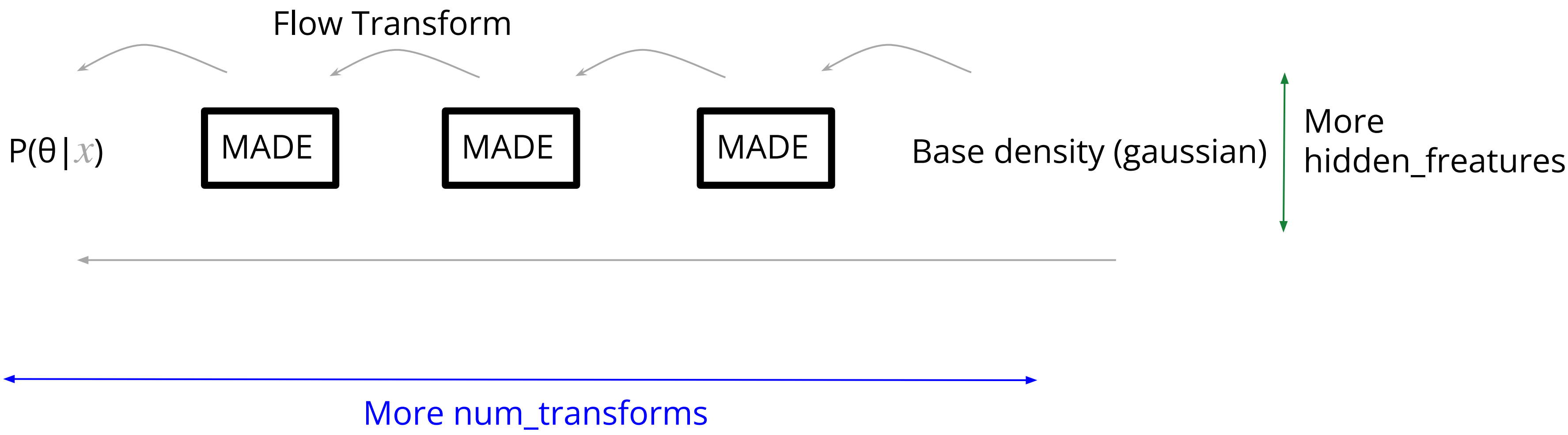
$$p(\mathbf{x}) = \pi_u\left(f^{-1}(\mathbf{x})\right)\left|\det\left(\frac{\partial f^{-1}}{\partial \mathbf{x}}\right)\right|$$

$$u_i = (x_i - \mu_i)\exp(-\alpha_i) \quad \text{where} \quad \mu_i = f_{\mu_i}(\mathbf{x}_{1:i-1}) \ \text{ and } \ \alpha_i = f_{\alpha_i}(\mathbf{x}_{1:i-1}).$$

$$\left|\det\left(\frac{\partial f^{-1}}{\partial \mathbf{x}}\right)\right| = \exp\left(-\sum_i \alpha_i\right) \quad \text{where} \quad \alpha_i = f_{\alpha_i}(\mathbf{x}_{1:i-1}).$$

Masked Autoregressive Flows are stacks of MADEs

NPE is constructed from stacks of MADE that are flow transformed.



Slide credit: Jason Poh

Embedding network to learn summary statistics

Simulator Input

parameters

Simulator output

images

Embedding network input

images

Embedding network output

Summary stats

Slide credit: Jason Poh

Overall trains to learn embedding network and NPE

Training an NPE + embedding

Simulator Input	Simulator output	Embedding network input	Embedding network output	NPE input	NPE output
parameters	images	images	Summary stats	[Summary stats] + [parameter set]	p(param|data)

Slide credit: Jason Poh

Masked Autoregressive Flows

There are mainly two families of neural density estimators that are both flexible and tractable:

- Normalizing flows
  - Transform a base density into target density by invertible transformation with tractable jacobian.
- Autoregressive models
  - Decomposes target density as a product of conditionals and models each conditional in turn.

This method is **both** - it is a normalizing flow of autoregressive models (Masked Autoencoders for Density Estimation (MADEs to be exact)

Neural Density Estimators

- An estimate of the exact posterior $p(\theta|x)$ can be learnt from neural density estimators $q_\phi(\theta|x)$
- Maximize the average log probability $\frac{1}{N}\sum_n log\ q_\phi(\theta_n | x_n)$

I think I'd like to get more into NPE because that's what we'll do in the tutorial. I'm planning on talking about normalizing flows and I"ll probably mention autoregressive flows, but not focus on them. I understand that a normalizing flow can estimate the likelihood, since we're sampling from x ~ p(x|theta), but I'm struggling with how to explain that it does this also for the posterior. I understand that it's accounting for the prior via drawing from it, but how is it accounting for the evidence?

Take a break

`sbi`: A toolkit for simulation-based inference

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

https://joss.theoj.org/papers/10.21105/joss.02505
https://github.com/sbi-dev/sbi

```python
from sbi.inference import infer
# import your simulator, define your prior over the parameters
parameter_posterior = infer(simulator, prior, method='SNPE', num_simulations=100)
```

Tutorial

- Linefit example:
  https://colab.research.google.com/drive/1CRBQqSim3KZV6s5hcwz-zMVmbJzTeaWz?usp=sharing

- Lenstronomy example:
  https://colab.research.google.com/drive/1NpwdTy98Ifo-vPul5Rt-HTFTalgP3cZH?usp=sharing

Fermilab

Flavors/algorithms of SNPE

SNPE-A: only allows Gaussian mixtures and mixture density models for modeling density

SNPE-B: more flexible but has some technical issues that are documented on github, having to do with the proposal distribution

SNPE-C: current state-of-the-art

Challenges / the future of SBI / State of the art / things that should keep you up at night

Challenges / the future of SBI / State of the art / things that should keep you up at night

General papers: https://simulation-based-inference.org/papers/

Applying to different areas for dramatic speed-up in inference: Khullar+2022

Fermilab

Challenges / the future of SBI / State of the art / things that should keep you up at night

General papers: https://simulation-based-inference.org/papers/

Applying to different areas for dramatic speed-up in inference: Khullar+2022

Differentiable simulators: Zhegal+2022

Calibration / a “crisis” in SBI?: Hermans+2021

SBI with many variables: Poh+2022

Guarantees on error?

Hierarchical

Challenges / the future of SBI / State of the art / things that should keep you up at night

General papers: https://simulation-based-inference.org/papers/

Applying to different areas for dramatic speed-up in inference: Khullar+2022

Differentiable simulators: Zhegal+2022

Calibration / a “crisis” in SBI?: Hermans+2021

SBI with many variables: Poh+2022

Guarantees on error?

Hierarchical

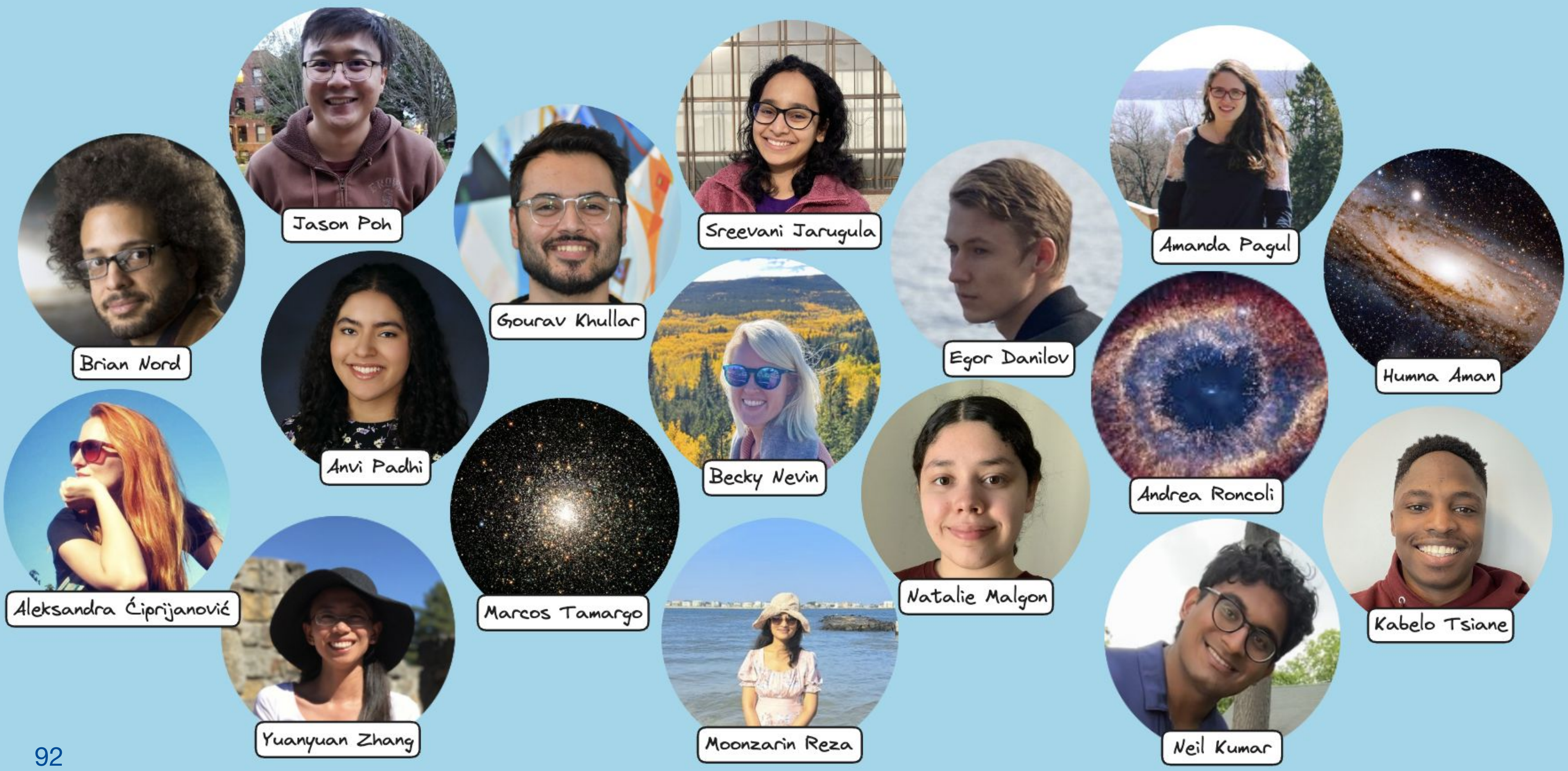
Graph neural nets / other architectures

Understanding really complicated simulations?

Domain adaptation!

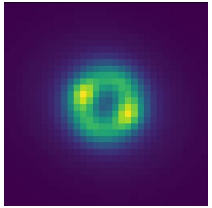
Fermilab

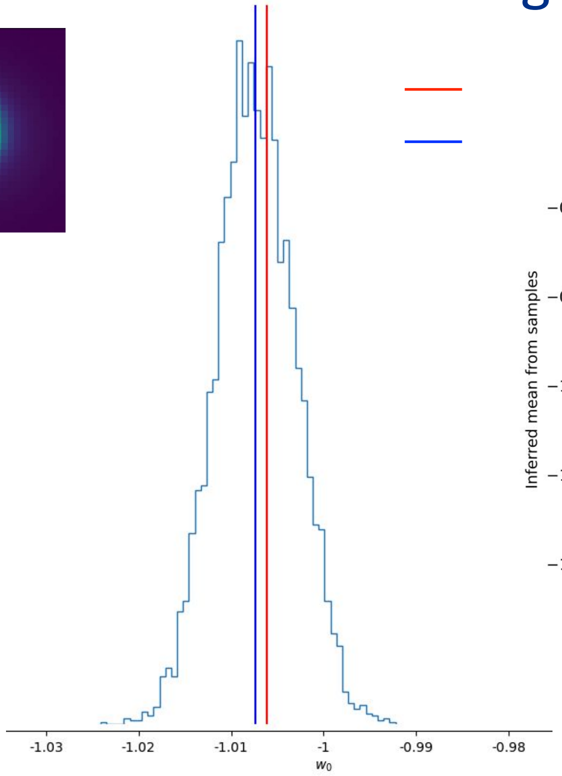
The Deepskies Lab SBI team



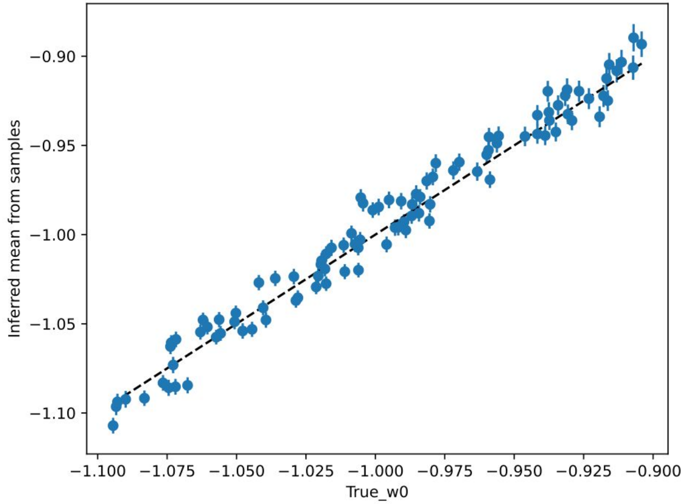
Dark Energy equation of state parameter

Inference from a single lens



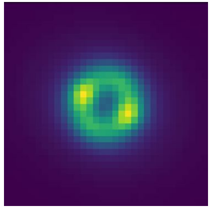


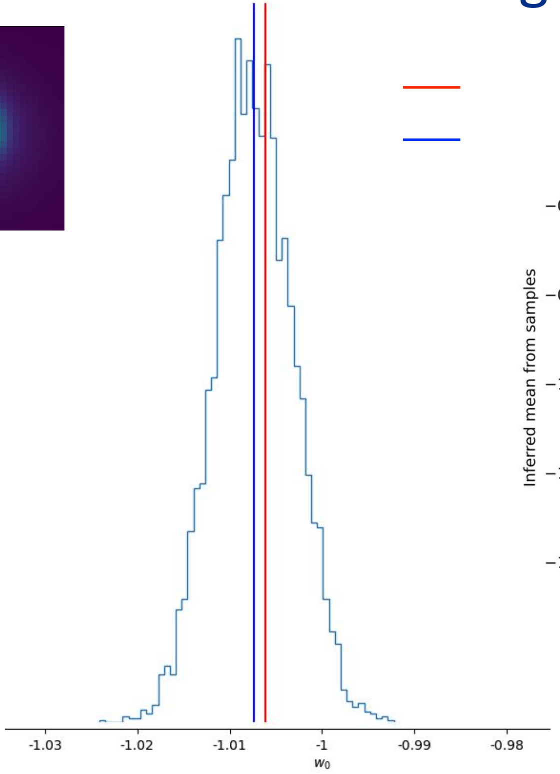
Inference from 100 lens



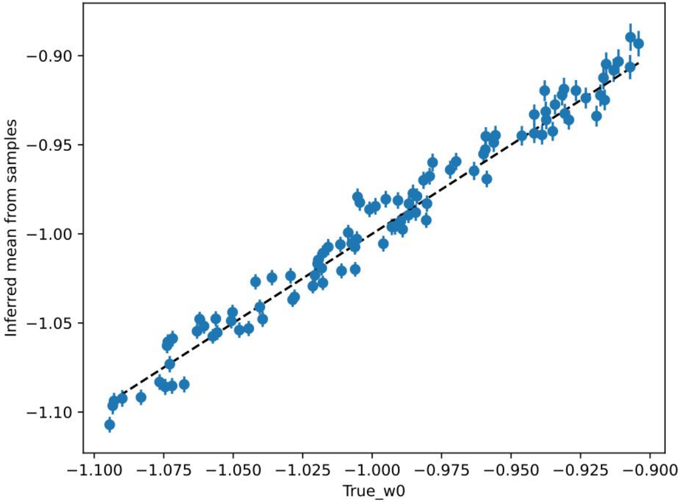
Dark Energy equation of state parameter

Inference from a single lens





Inference from 100 lens



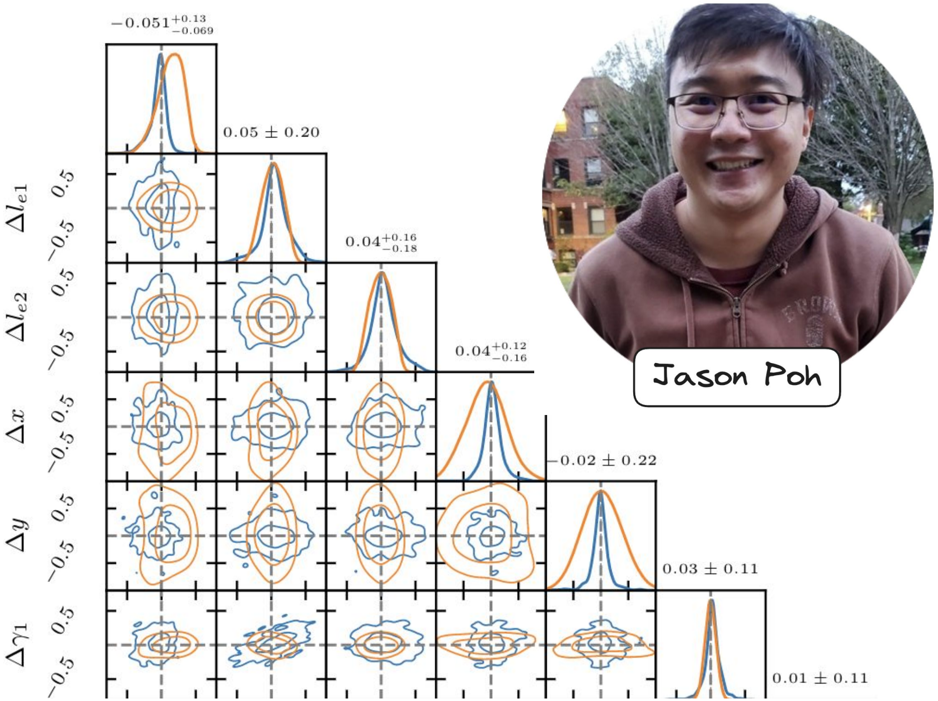
Next steps:

- astrophysics + cosmology
- Joint inference

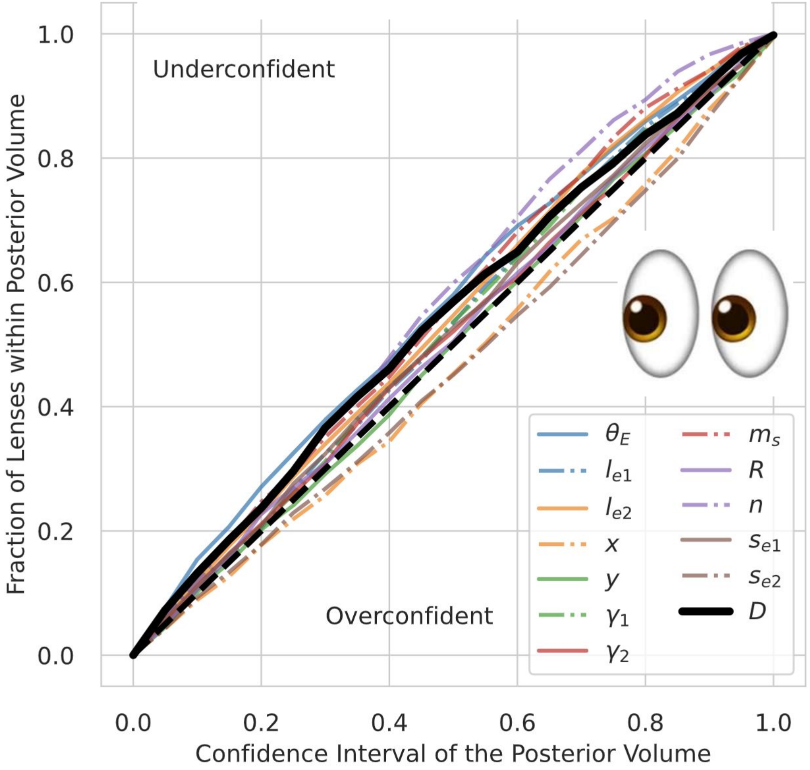
Fermilab

SBI to infer astrophysics parameters from strong lensing

Jason Poh+ and DeepSkies



Posterior coverage plot!



SBI to infer galaxy properties from spectra

Khullar et al. 2022 and DeepSkies

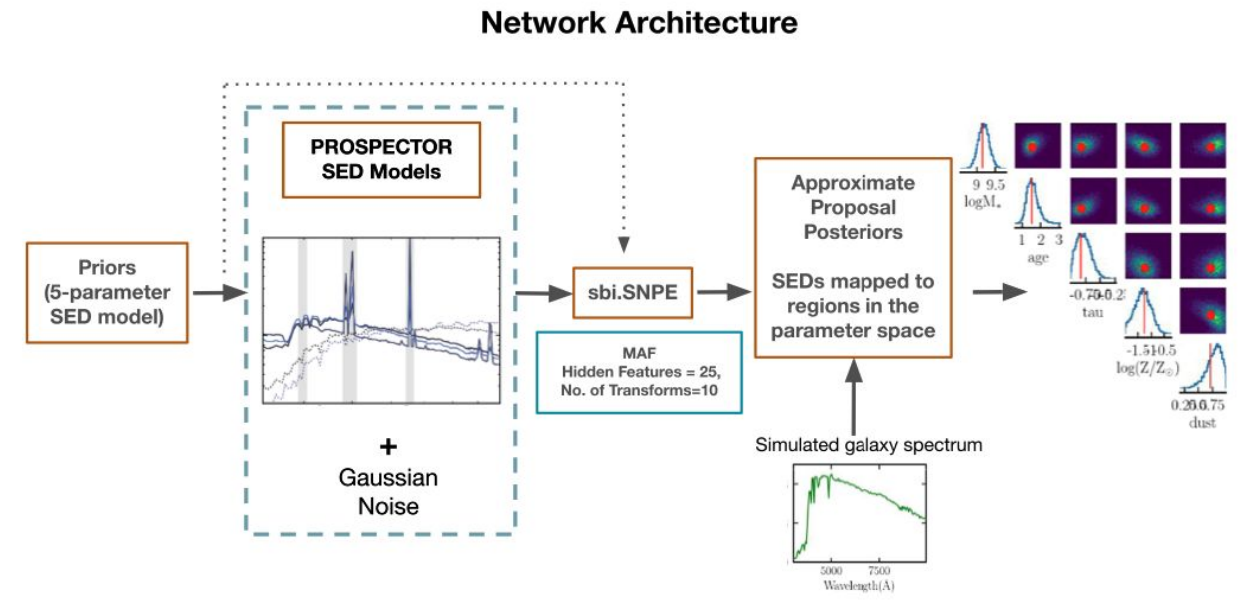
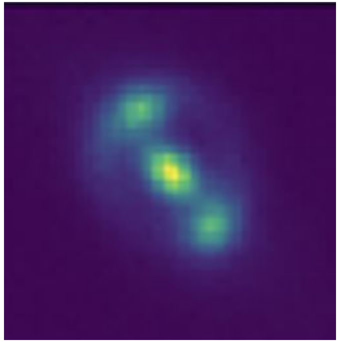
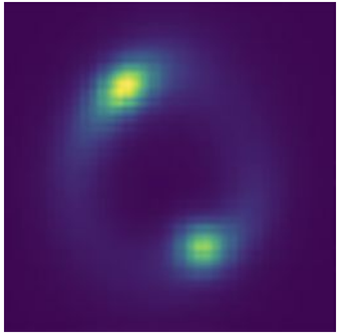


Figure 2. The architecture used in this work to infer galaxy SED properties with spectroscopic data. We use a five-parameter model and a training set with realistic spectra, that is trained by an NPE to generate approximate posteriors.

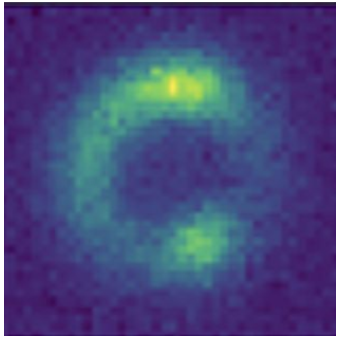
Domain adaptation to improve the performance of SBI on real observational lenses with noise!

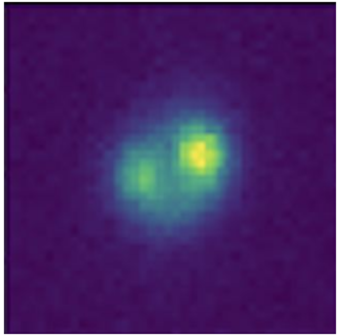
Source domain:
low noise simulated lenses





Target domain:
DES noise simulated lenses



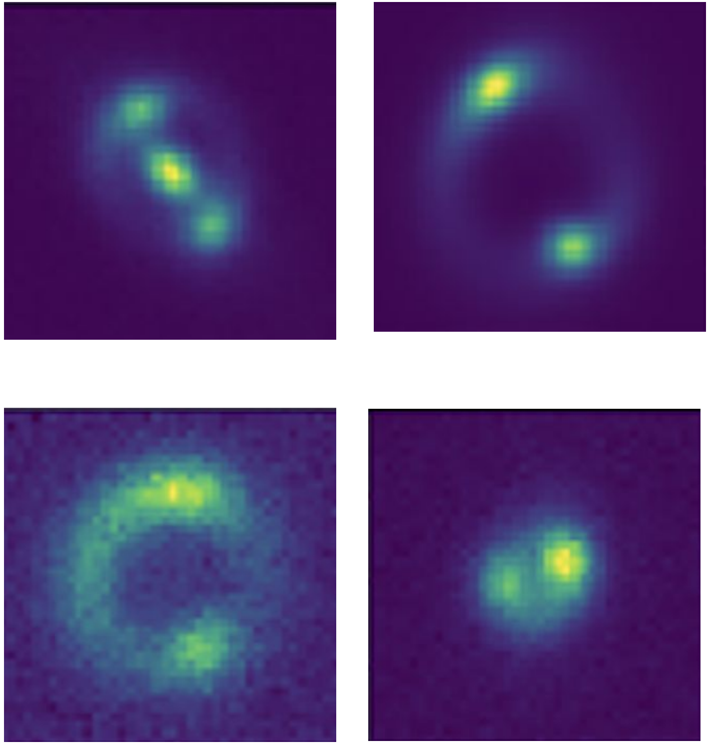


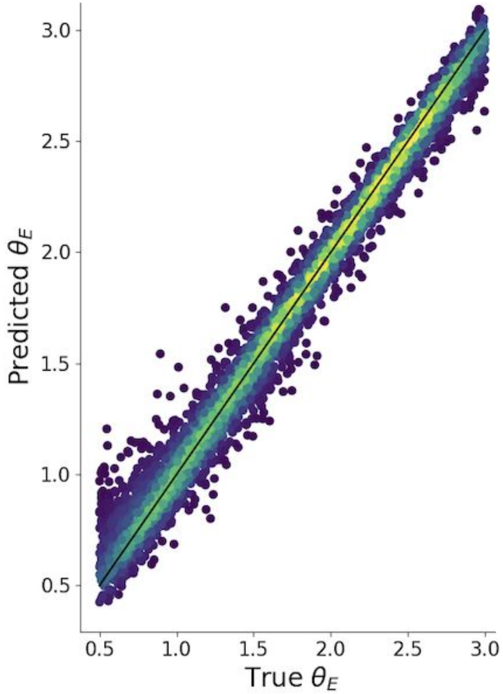
Fermilab

Domain adaptation to improve the performance of SBI on real observational lenses with noise!

Source domain: low noise simulated lenses

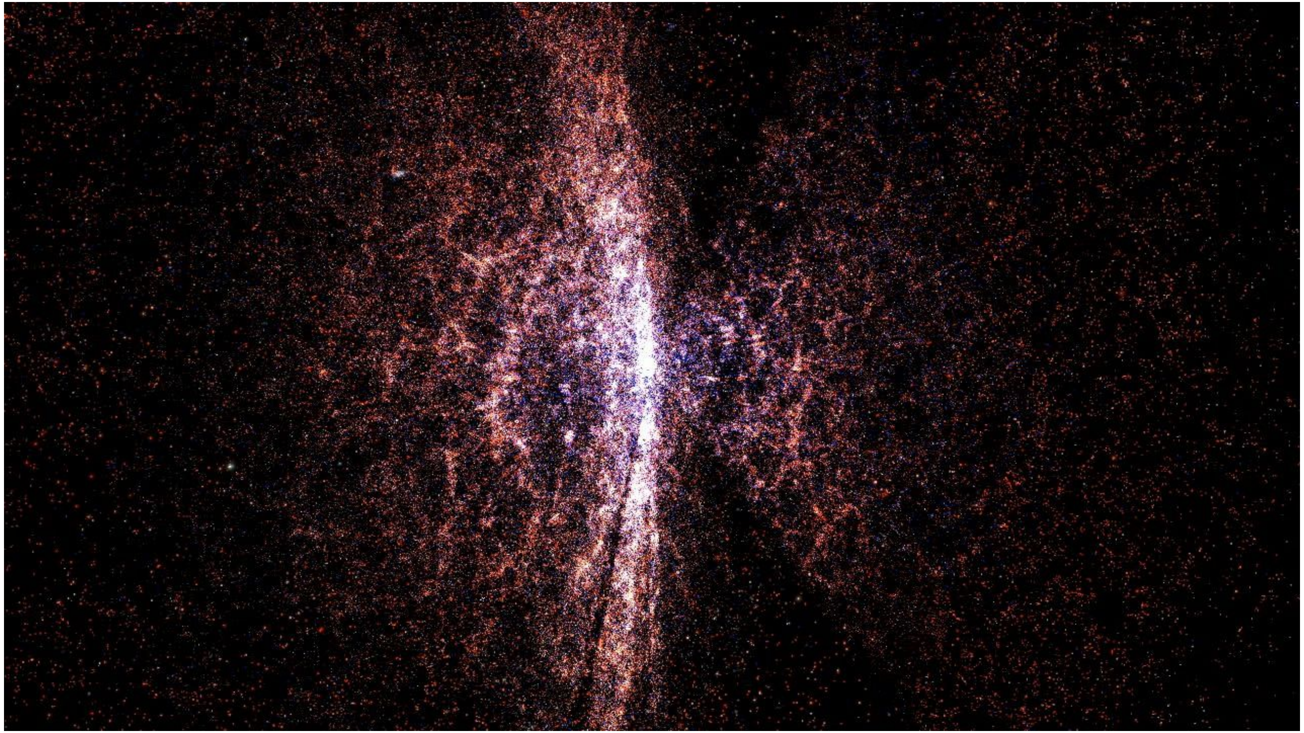
Target domain: DES noise simulated lenses





SBI for alternate networks, like graph neural networks

Goal: Infer key cosmological parameters σ_8 and Ω_{matter}



If you take anything from this let it be:

- Simulation-based Inference is an alternative to likelihood-based inference
  - Does not require the computation of a likelihood
  - Machine learning methods are amortized, i.e. can evaluate posterior from new data without retraining the model
  - Computationally more efficient than MCMC based methods

If you take anything from this let it be:

- Simulation-based Inference is an alternative to likelihood-based inference

- Lots of options for what type of probability you want to target (likelihood, likelihood ratio, posterior)

- The future is exciting, lots of research here into expanding into new data types, looking at uncertainty guarantees, lots of opportunities, especially for those interested in explainability, ie the HEP community

Join the Deepskies Lab!

https://deepskieslab.com/

Google form to join.

The end

If you take anything from this let it be:

- Simulation-based Inference is an alternative to likelihood-based inference
- Lots of options for what type of probability you want to target (likelihood, likelihood ratio, posterior)
- The future is exciting, lots of research here into expanding into new data types, looking at uncertainty guarantees, lots of opportunities, especially for those interested in explainability, ie the HEP community

EXTRAS

Notes from Jason's SBI session

- Where to not use SBI - if your simulator is not good, doesn't capture actual physics?
  - Activation energy of it being worth it → if MCMC is quick enough don't need to use it, it has additional upfront cost,
- How complex model can be before you switch?
  - Sam: curse of dimensionality, SBI can do 20 or so parameters, also embedding network
  - Complexity of likelihood versus complexity of parameters

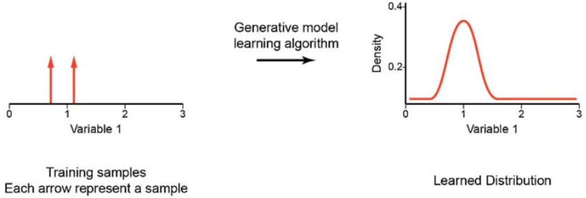
Density Estimators (Thanks to Jason!)

- Unsupervised method of getting structure from data:
  - Given , what is ?
  - Given , what is ?
- Gaussian Density Model

Model: $$q_{\phi}(\mathbf{x}) = \frac{1}{|\det(2\pi\boldsymbol{\Sigma})|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) \quad \text{where} \quad \boldsymbol{\phi} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}.$$

Training: Parametric density models are typically estimated by maximum likelihood. Given a set of training datapoints $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ that have been independently and identically generated by a process with density $p(\mathbf{x})$, we seek a setting of the model's parameters $\boldsymbol{\phi}$ that maximize the average log likelihood on the training data:

$$L(\boldsymbol{\phi}) = \frac{1}{N}\sum_n \log q_{\boldsymbol{\phi}}(\mathbf{x}_n). \tag{2.13}$$



From [Papamakarios (2019)]

Neural Density Estimators

- Neural Networks to parameterize density model

$$L(\phi) = \frac{1}{N}\sum_n \log q_\phi(\mathbf{x}_n) = \frac{1}{N}\sum_n f_\phi(\mathbf{x}_n).$$

- The parameters of the Neural network are updated through gradient descent

$$\nabla_\phi \hat{L}(\phi) = \frac{1}{M}\sum_m \nabla_\phi f_\phi(\mathbf{x}_{n_m}).$$

- SBI has 4 built-in density estimators:
  - Masked Autoregressive Flow (MAF)
  - Neural Spline Flow (NSF)
  - Masked Autoencoder for Distribution Estimation (MADE)
  - Mixture Density Network (MDN)