Parameter inference for binary black holes using deep learning

Stephen R. Green
Albert Einstein Institute Potsdam

(based on arXiv:2008.03312 with J. Gair)

ICERM Workshop on Statistical Methods for the Detection, Classification, and Inference of Relativistic Objects

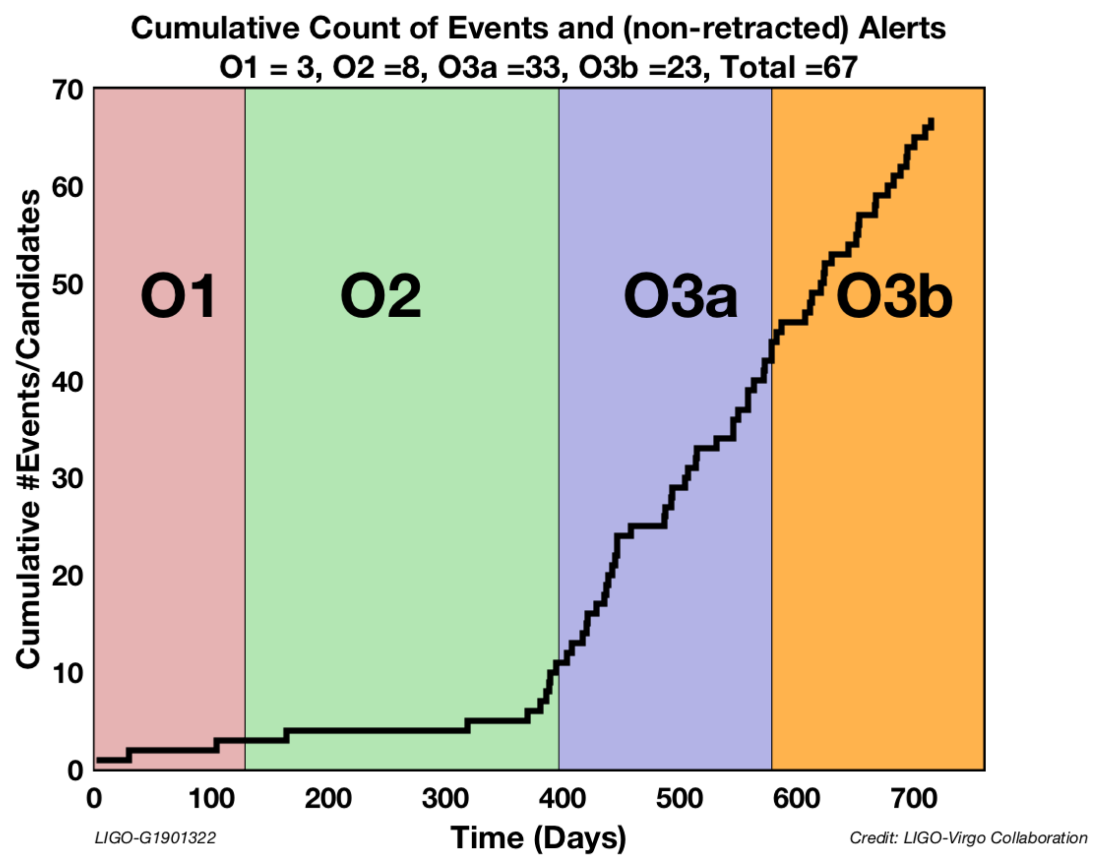
November 15, 2020

Outline

1. Introduction: Bayesian inference with iterative samplers
2. Simulation-based inference with normalizing flows
3. Application to binary black hole parameter estimation
4. Demonstration on GW150914
5. Outlook

Introduction

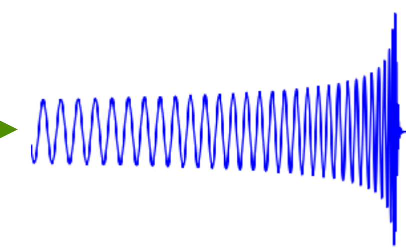
- Since the first detection of gravitational waves, there have been steady improvements in detector sensitivity.
    - 50 published detections of compact binaries
    - Two binary neutron stars, one with multi-messenger counterpart
- Enabled tests of gravity, understanding of neutron-star physics, and placed constraints on binary populations and cosmology.



Introduction

$$R_{ab} - \frac{1}{2} R g_{ab} = 8\pi T_{ab}$$

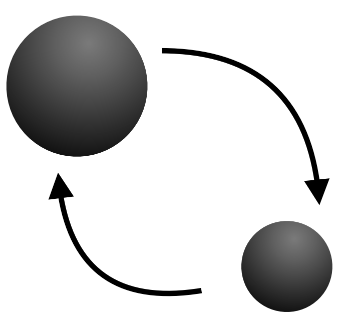
Physics modeling

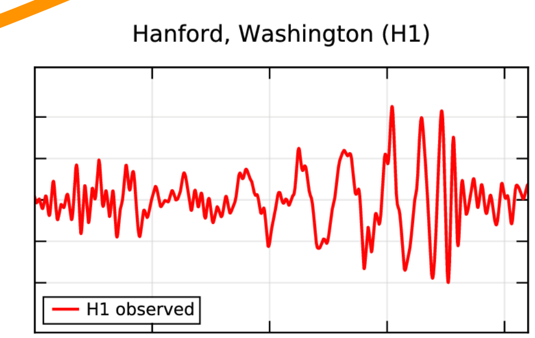


$h(\theta)$

Astrophysical systems

Detector data

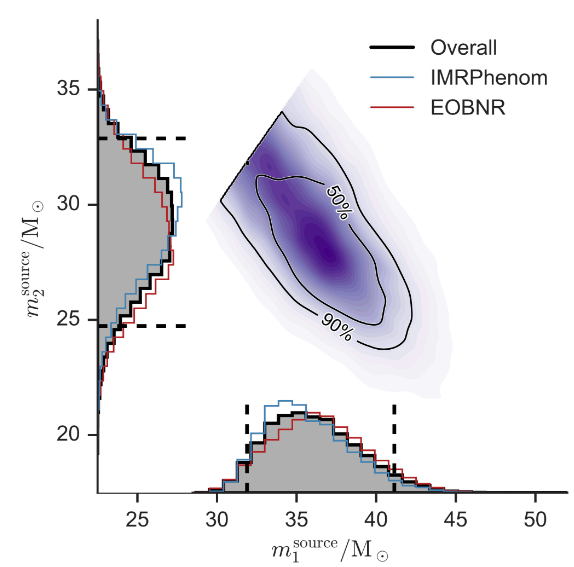




Bayesian inference

parameters θ

$$p(\theta|s) = \frac{p(s|\theta)p(\theta)}{p(s)}$$



strain s

Bayesian parameter inference for compact binaries

- Sample posterior distribution for system parameters θ (masses, spins, sky position, etc.) given detector strain data s.

likelihood

prior

$$p(\theta \mid s) = \frac{p(s \mid \theta) p(\theta)}{p(s)}$$

evidence (normalizing factor)

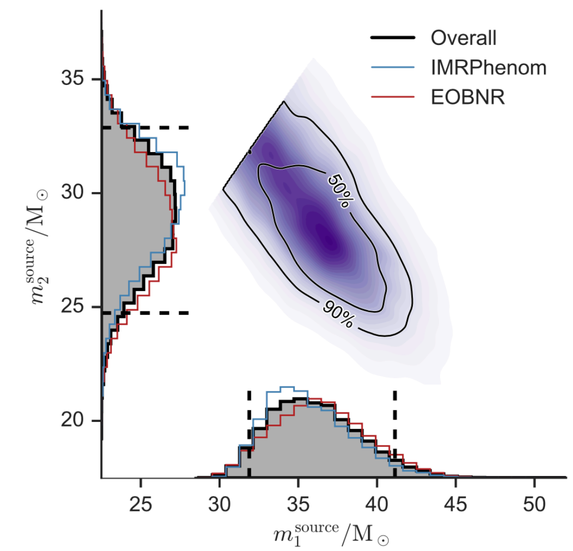


Image: Abbott et al (2016)

- Likelihood based on assumption of stationary Gaussian detector noise

$$p(s \mid \theta) \propto \exp\left(-\frac{1}{2} \sum_I \left(s_I - h_I(\theta) \mid s_I - h_I(\theta) \right) \right)$$

waveform model

where $\quad \left(a \mid b \right) = 2 \int_0^\infty \mathrm{d}f \, \frac{\hat{a}(f)\hat{b}(f)^* + \hat{a}(f)^*\hat{b}(f)}{S_n(f)}$

Bayesian parameter inference for compact binaries

- Sample posterior distribution for system parameters θ (masses, spins, sky position, etc.) given detector strain data s.

likelihood

prior

$$p(\theta \mid s) = \frac{p(s \mid \theta)p(\theta)}{p(s)}$$

evidence (normalizing factor)

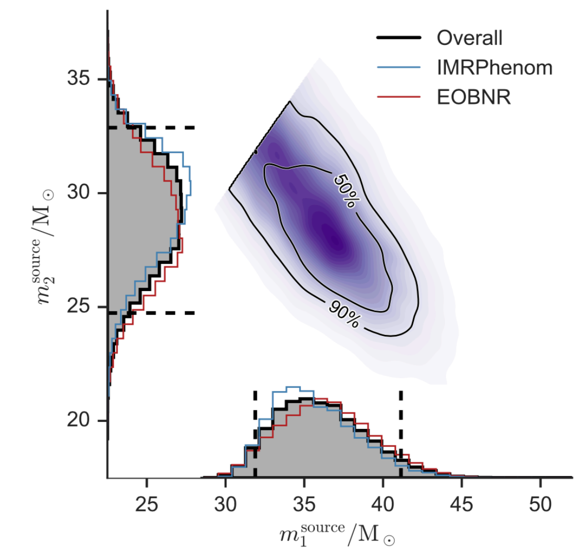


Image: Abbott et al (2016)

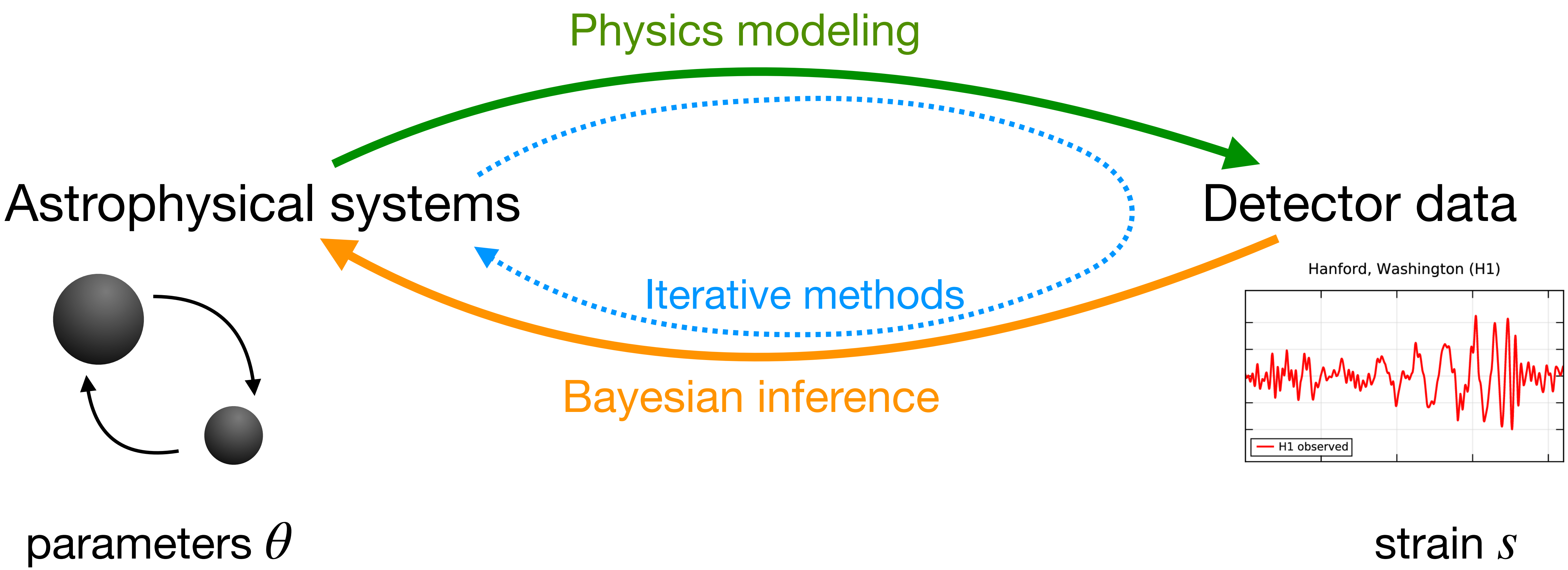
- Prior $p(\theta)$ based on beliefs about system before looking at data,

  e.g., uniform in m_1, m_2 over some range,
  uniform in spatial volume,
  etc.

- Once likelihood and prior are defined, posterior can be evaluated up to normalization.

Introduction

- To obtain samples $\theta \sim p(\theta \,|\, s)$, typically use an iterative method, such as Markov chain Monte Carlo (MCMC) or nested sampling.

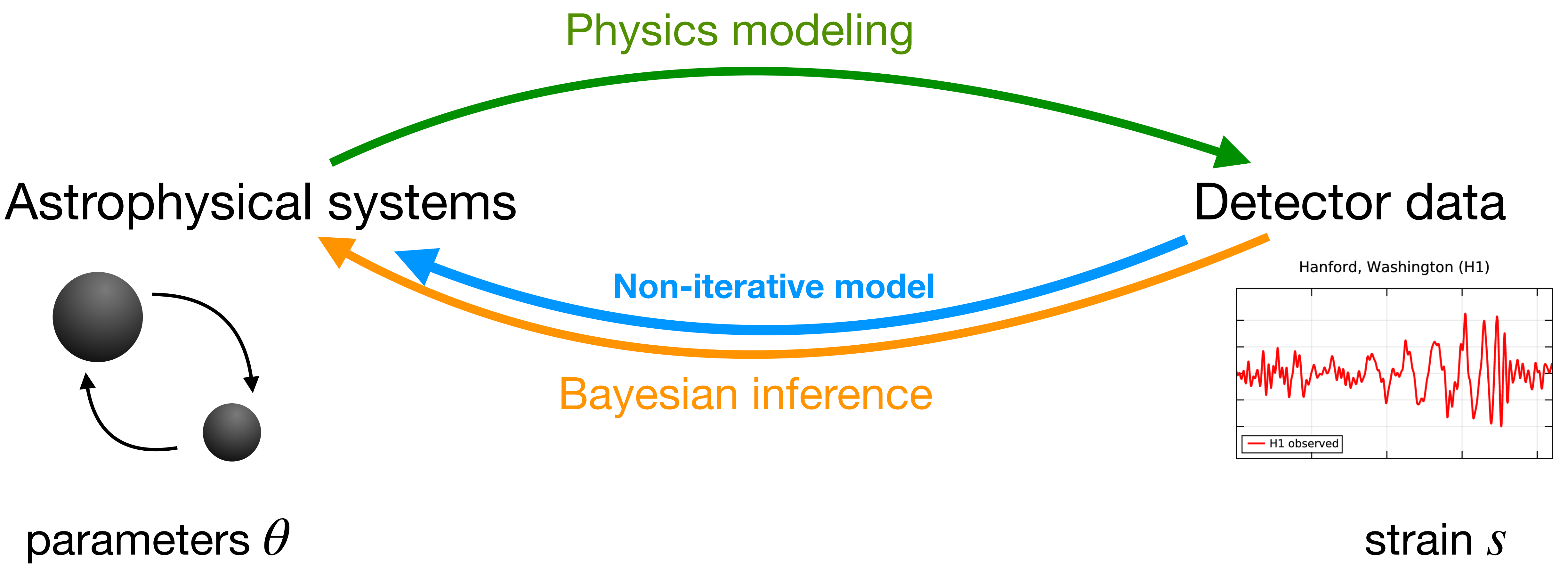


Iterative samplers

- **Computationally expensive:**
  - Many likelihood evaluations required for each independent sample.
  - Likelihood evaluation slow, requires a waveform to be generated.
  - Days to weeks for inference for a single event, depending on type of event and waveform model. Fast inference needed for multi-messenger followup.
  - Inference must be repeated for every event. Detection rate growing with detection sensitivity.
- **Limited scope:**
  - Requires ability to evaluate likelihood. Noise must be (stationary) Gaussian.

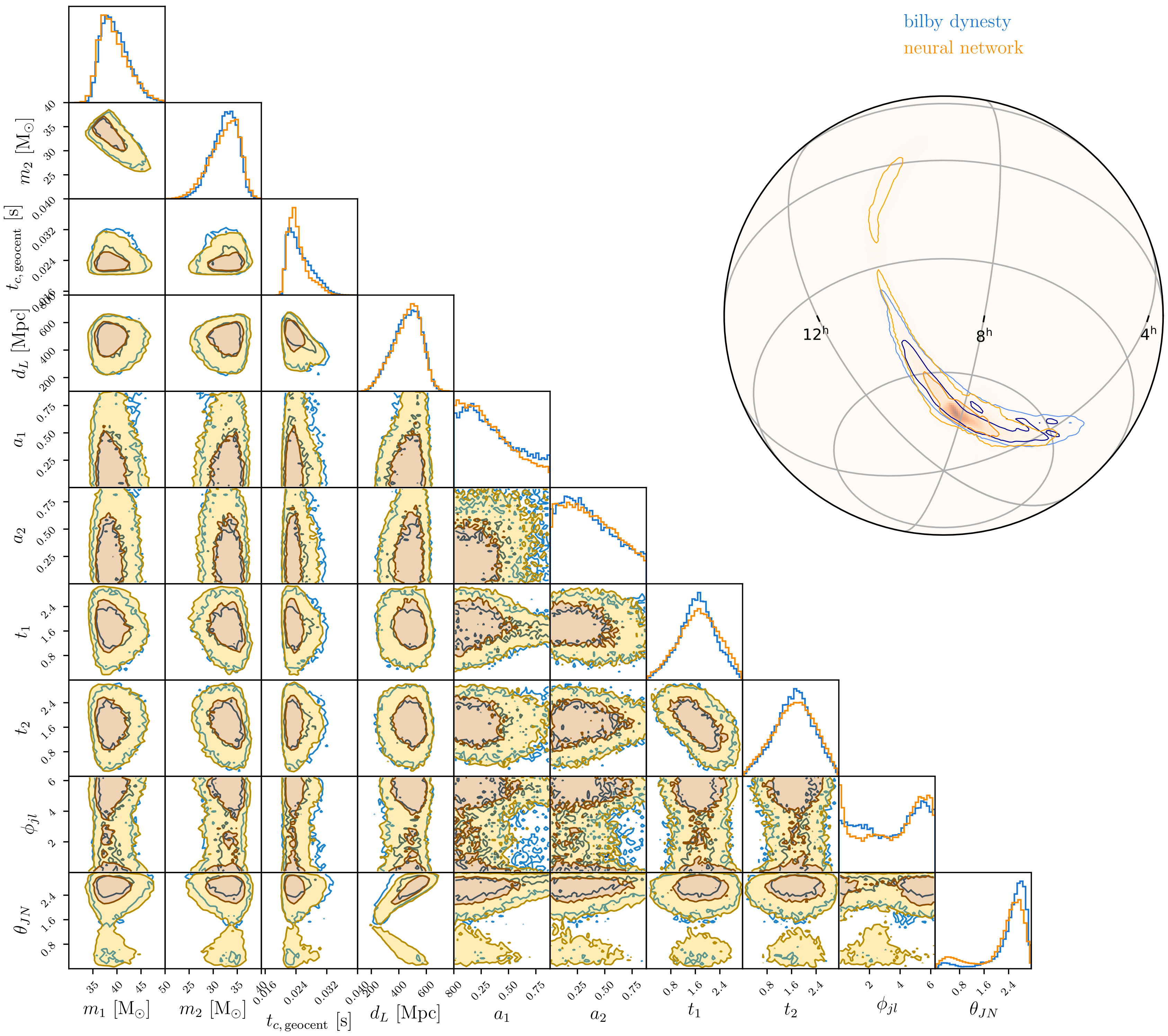
Introduction

- **Can we build a non-iterative inverse model?**




YES WE CAN

Demonstration on GW150914



Rest of this talk:
How to do this?

Two key ideas

1. **Neural-network conditional density estimator $q(\theta \,|\, s)$**:

   - Represent complicated distributions using method of normalizing flows.

   - Fast sampling and evaluation.

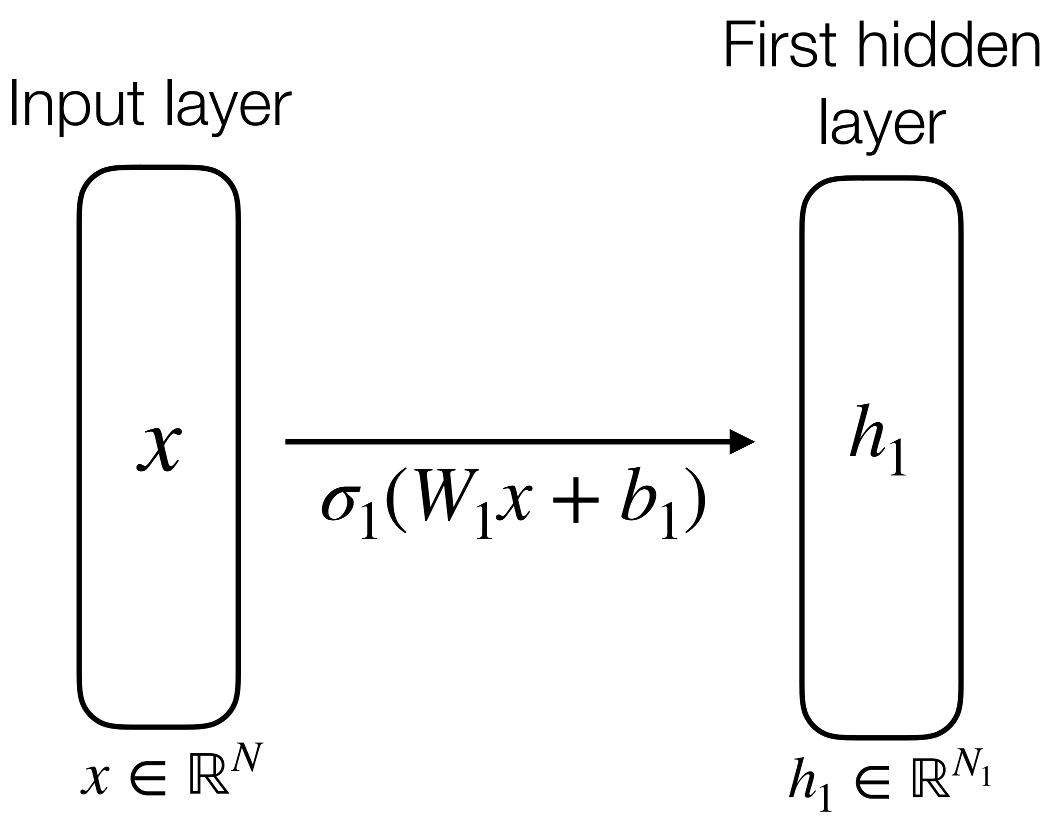
2. **Simulation-based inference**:

   - Training $q(\theta \,|\, s) \to p(\theta \,|\, s)$ requires only simulated data $s \sim p(s \,|\, \theta)$.

   - No posterior samples or likelihood evaluations.

Introduction to neural networks

- Nonlinear functions as composition of simple mappings:



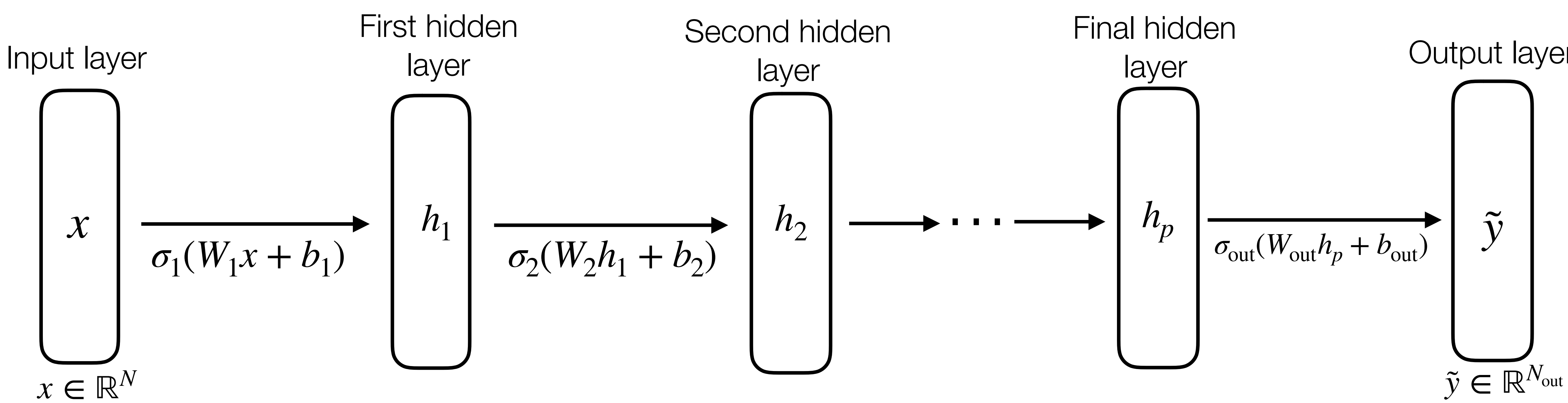
1. Affine transformation

$$W_1 x + b_1$$

2. Element-wise nonlinear mapping

$$\sigma_{1,i}(x_i) = \begin{cases} x_i & \text{if } x_i \geq 0, \\ 0 & \text{if } x_i < 0. \end{cases}$$

Introduction to neural networks



- (x, y) pairs of training data $\longrightarrow$ learn a function $y(x)$
- Minimize loss function, e.g., $L = \mathbb{E}_{\{(x,y)\}} \sum_{i=1}^{N_{\text{out}}} (\tilde{y}_i(x) - y_i)^2$
- Tune (W_i, b_i) using stochastic gradient descent.

Neural networks as probability distributions

- Interpret the neural network as a conditional probability distribution.

$$\text{function } \tilde{y}(x) \qquad \longrightarrow \qquad \text{distribution } q(y|x)$$
$$= \mathcal{N}(\tilde{y}(x), \mathbb{1})(y)$$
$$= \frac{1}{(2\pi)^{N_{\text{out}}/2}} \exp\left(-\frac{1}{2}\sum_{i=1}^{N_{\text{out}}}(y_i - \tilde{y}_i(x))^2\right)$$

- Maximize the likelihood that $\{(x, y)\}$ came from $q(y\,|\,x)$,

$$L = \mathbb{E}[-\log q(y|x)] \propto \mathbb{E}\left[\sum_{i=1}^{N_{\text{out}}}(y_i - \tilde{y}_i(x))^2\right]$$

Squared difference loss!

- More complicated distributions can also be parametrized by neural networks.

Simulation-based inference

[First applied to GW by Chua and Vallisneri (2020), Gabbard et al (2019)]

- Train network to model true posterior, as given by prior and likelihood that we specify, i.e.,

$$q(\theta \,|\, s) \to p(\theta \,|\, s)$$

- Minimize expectation value (over s) of cross-entropy between the distributions

$$L = -\int \mathrm{d}s\, p(s) \int \mathrm{d}\theta\, p(\theta \,|\, s)\, \log q(\theta \,|\, s)$$

Intractable with knowing posterior for each s!

- Bayes' theorem $\implies$ $p(s)\, p(\theta \,|\, s) = p(\theta)\, p(s \,|\, \theta)$

$$\boxed{\therefore\ L = -\int \mathrm{d}\theta\, p(\theta) \int \mathrm{d}s\, p(s \,|\, \theta)\, \log q(\theta \,|\, s)}$$

Only requires samples from likelihood, not the posterior!

Simulation-based inference

- Loss function

$$\boxed{\begin{aligned} L &= -\int \mathrm{d}\theta\, p(\theta) \int \mathrm{d}s\, p(s|\theta) \log q(\theta|s) \\ &\approx -\frac{1}{N}\sum_{i=1}^{N} \log q(\theta^{(i)}|s^{(i)}), \qquad \text{where } \theta^{(i)} \sim p(\theta),\ s^{(i)} \sim p(s|\theta^{(i)}) \end{aligned}}$$

Estimate on minibatch of size N

Easy to evaluate from neural network

Sample strain data from generative process (likelihood)

Sample parameters from prior

- Choose network parameters that minimize L: compute gradient of L with respect to network parameters and use stochastic gradient descent.

- Never evaluate a likelihood and no need for posterior samples!

Gravitational-wave parameter estimation

- Chua and Vallisneri (2020) applied SBI with Gaussian $q(\theta \mid s)$ to gravitational waves:

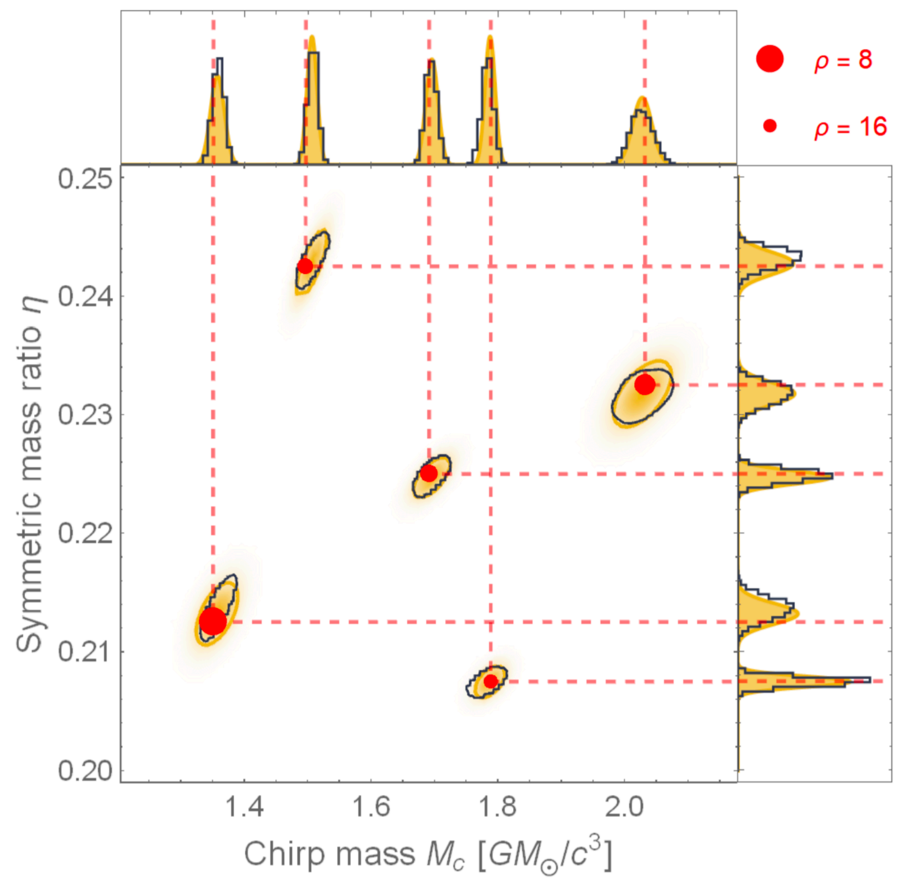


Figure: Chua and Vallisneri (2020)

- Works for high signal-to-noise, but more generally distributions can have higher moments and multimodality.

- Require $q(\theta \mid s)$ with flexible distribution over θ and complicated dependence on s.

Conditional density estimator

- **Our approach**: Model defined by a normalizing flow $f_s : u \mapsto \theta$ from a simple distribution to a complex one:

1. f_s invertible

2. simple Jacobian determinant

$$q(\theta \,|\, s) = \pi\left(f_s^{-1}(\theta)\right) \left|\det J_{f_s}^{-1}\right|$$

Much more complicated distribution

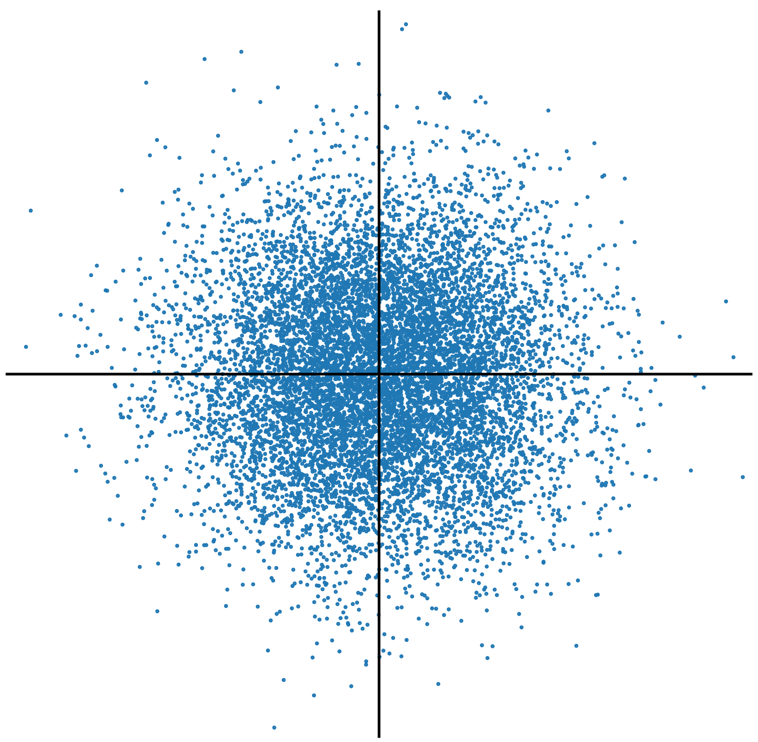
Multivariate standard normal
$\mathcal{N}(0,1)^d$

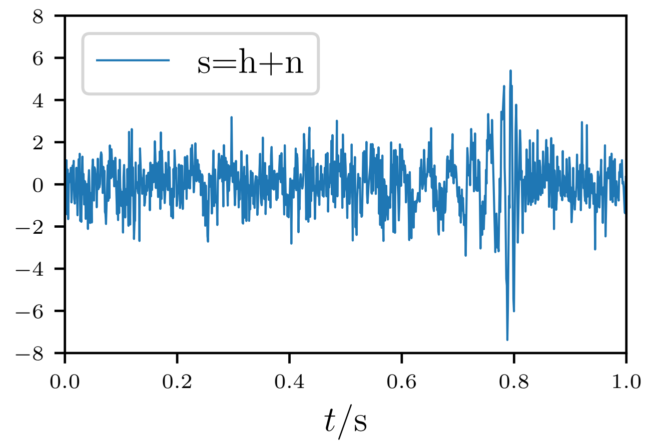
- Easy to sample and evaluate $\pi(u) \implies$ same for $q(\theta \,|\, s)$.

- Define normalizing flow in terms of a neural network.

Normalizing flows for gravitational waves

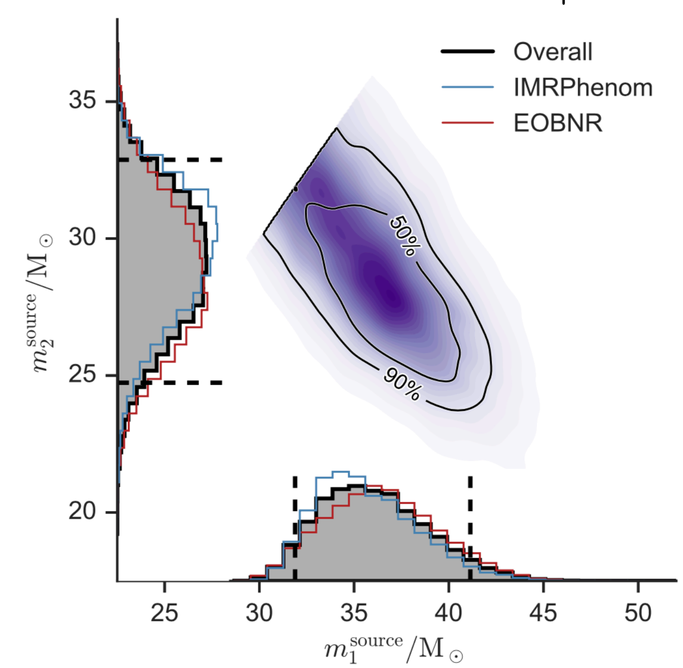
$u \sim \mathcal{N}(0,1)^D$





$$\theta = f_s(u)$$

$$\theta \sim q(\theta|s)$$
$$= \mathcal{N}(0,1)^D(f_s^{-1}(\theta)) \left|\det J_{f_s}^{-1}\right|$$



(hopefully)

Normalizing flow

- Requirements:

    1. Invertible ✓

    2. Simple Jacobian determinant ✓ $$\det J_{f_s} = \prod_{i=d+1}^{D} c_i'(u_i; u_{1:d}, s)$$

- Use a sequence of “coupling transforms”:

$$c_{s,i}(u) = \begin{cases} u_i & \text{if } i \leq d, \\ c_i(u_i; u_{1:d}, s) & \text{if } i > d. \end{cases}$$

Hold fixed half of the components

Transform remaining components element-wise, conditional on other half and s.

- c_i should be differentiable and have analytic inverse with respect to u_i.

Normalizing flow

- Neural spline flow (Durkan et al, 2019):

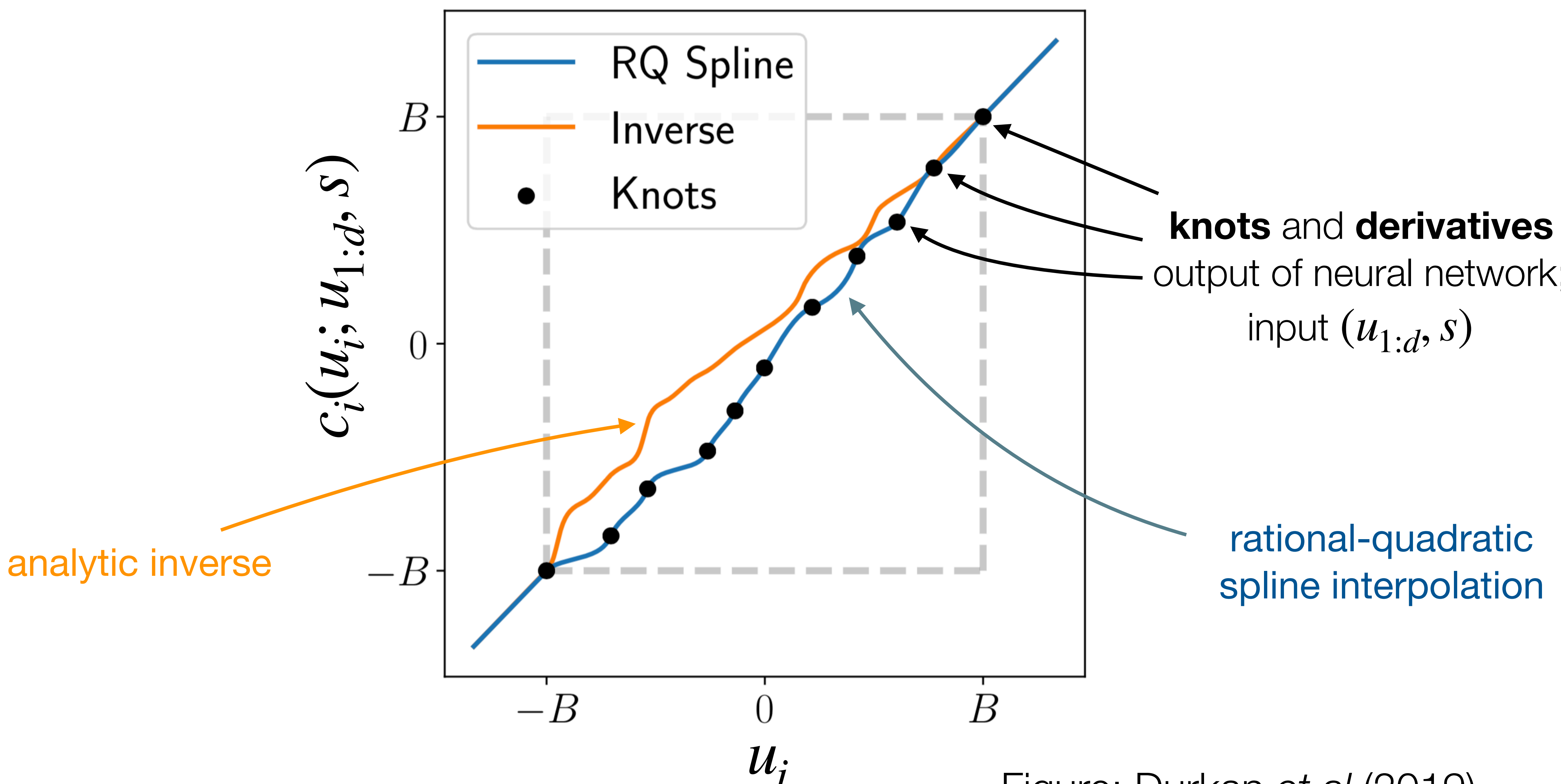


Figure: Durkan *et al* (2019)

Normalizing flow

Neural spline flow can represent very complicated multimodal distributions:

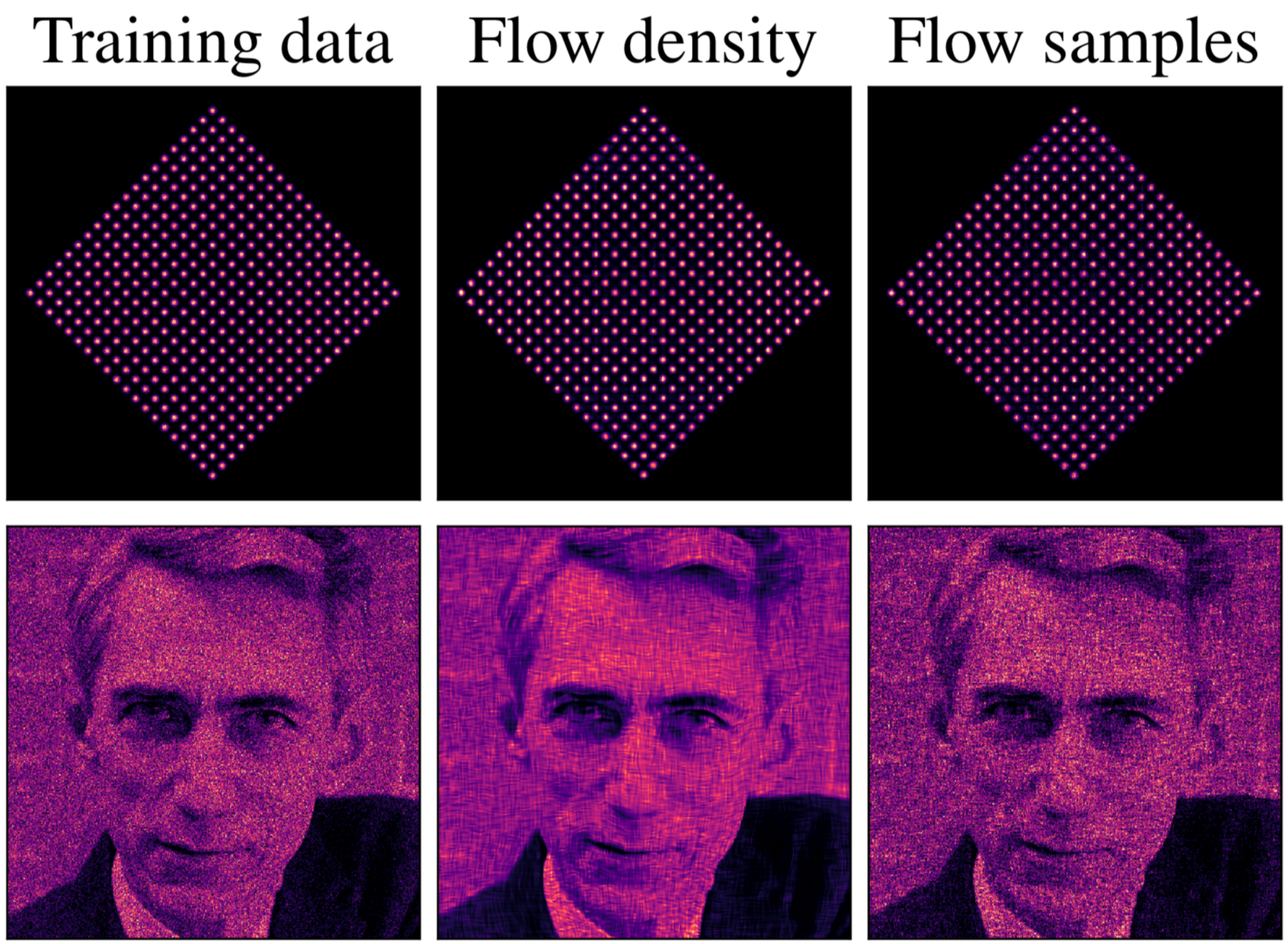


Image: Durkan *et al* (2019)

Normalizing flow

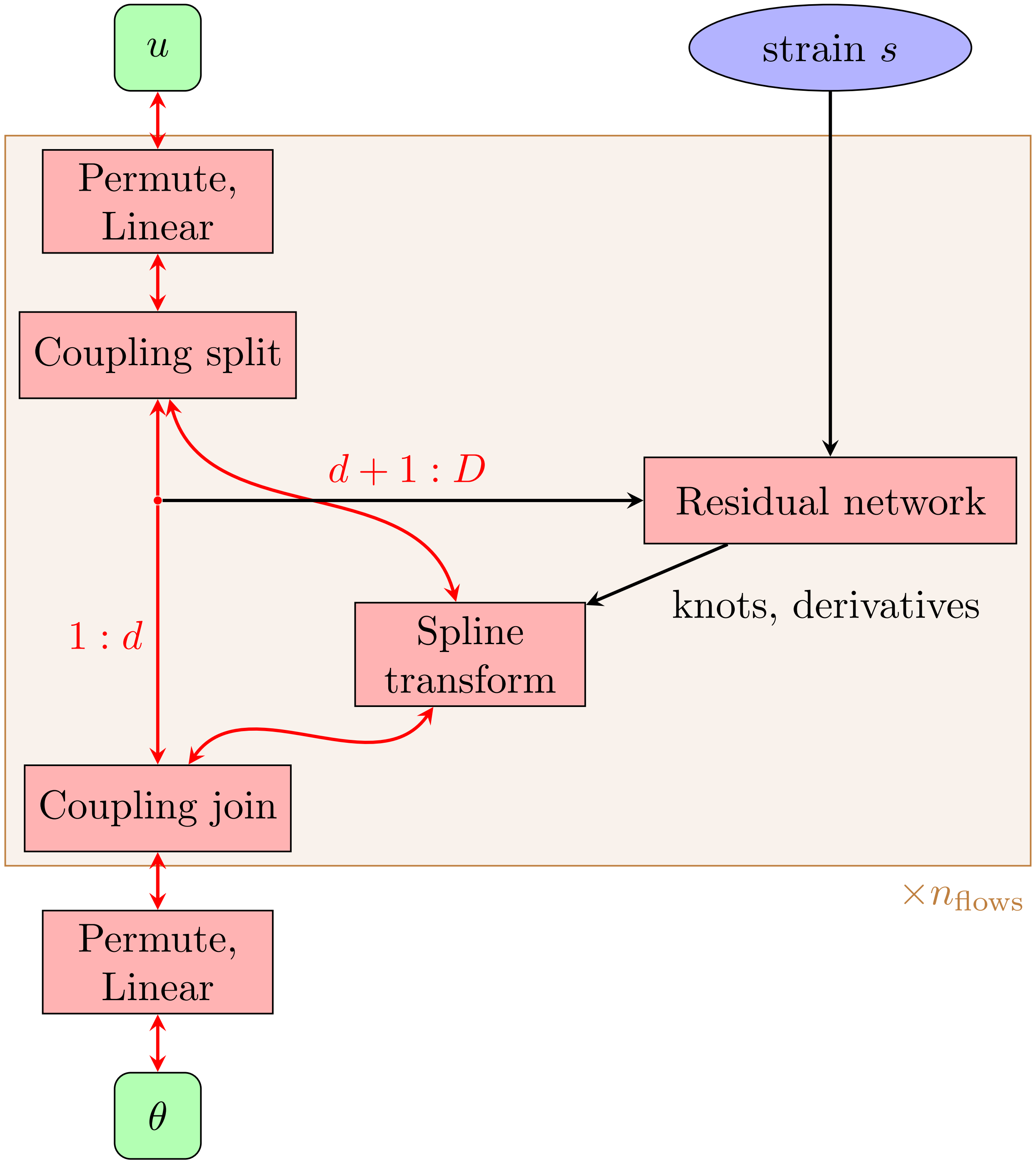
- Transform half the components in each coupling transform

$$c_{s,i}(u) = \begin{cases} u_i & \text{if } i \le d, \\ c_i(u_i; u_{1:d}, s) & \text{if } i > d. \end{cases}$$

Rational-quadratic spline function

- parametrized by functions of $(u_{1:d}, s)$
- analytic inverse and derivative

- Sequence of transformations give very flexible $q(\theta \mid s)$.



Application to binary black holes

- Recall loss function

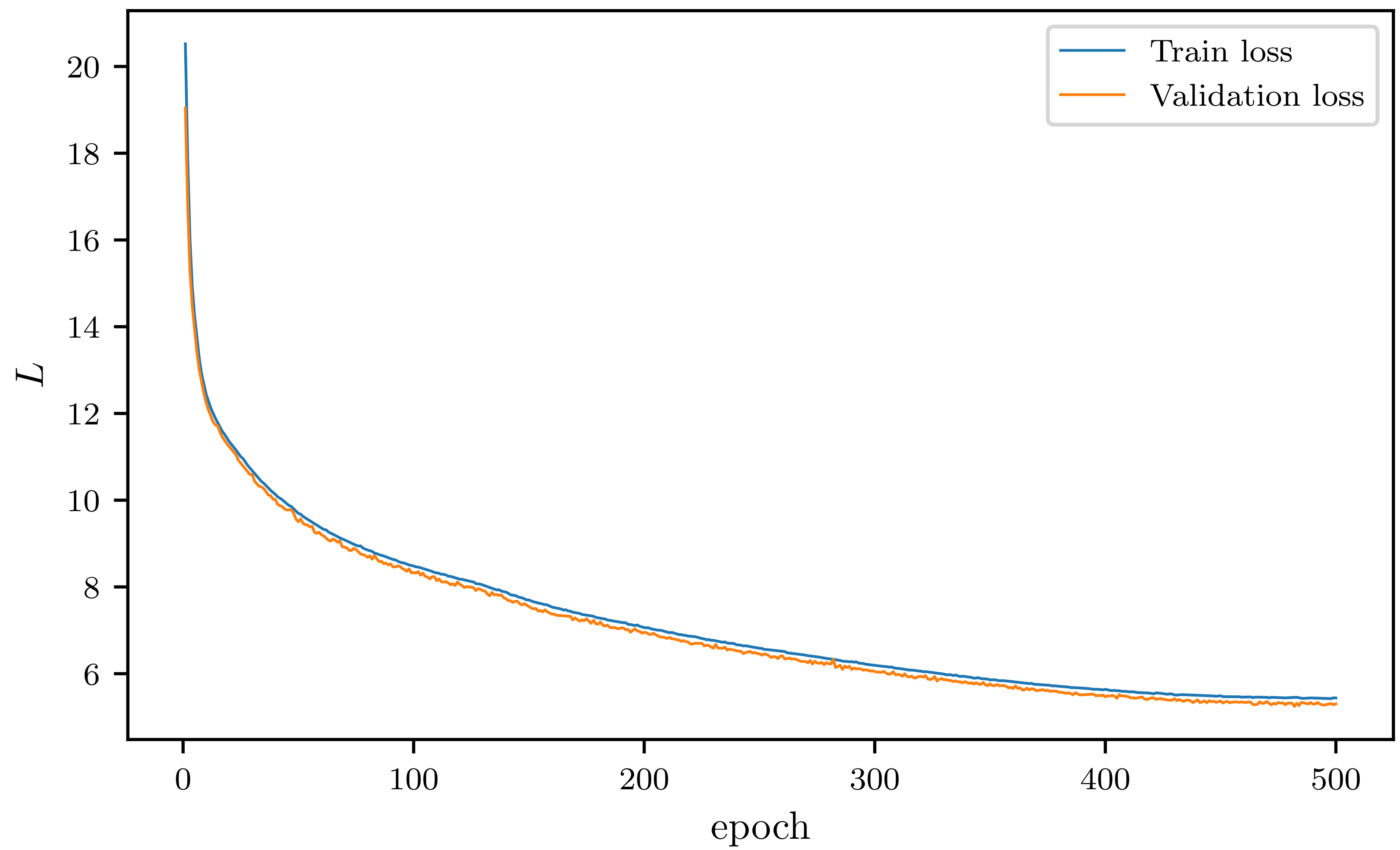
$$L \approx -\frac{1}{N}\sum_{i=1}^{N} \log q\left(\theta^{(i)}|s^{(i)}\right), \qquad \text{where } \theta^{(i)} \sim p(\theta) \text{ and } s^{(i)} \sim p(s|\theta^{(i)})$$

- Training requires simulated data.

1. Draw parameters from prior, $\theta^{(i)} \sim p(\theta)$ — 15D space for binary black holes
2. Calculate waveform using a model, $h^{(i)} = h(\theta^{(i)})$ — IMRPhenomPv2
3. Add stationary Gaussian noise, $s^{(i)} = h^{(i)} + n^{(i)}$, where $n^{(i)} \sim p_S(n)$. — PSD at time of event
4. Calculate $q\left(\theta^{(i)} \,|\, s^{(i)}\right)$ using normalizing flow.

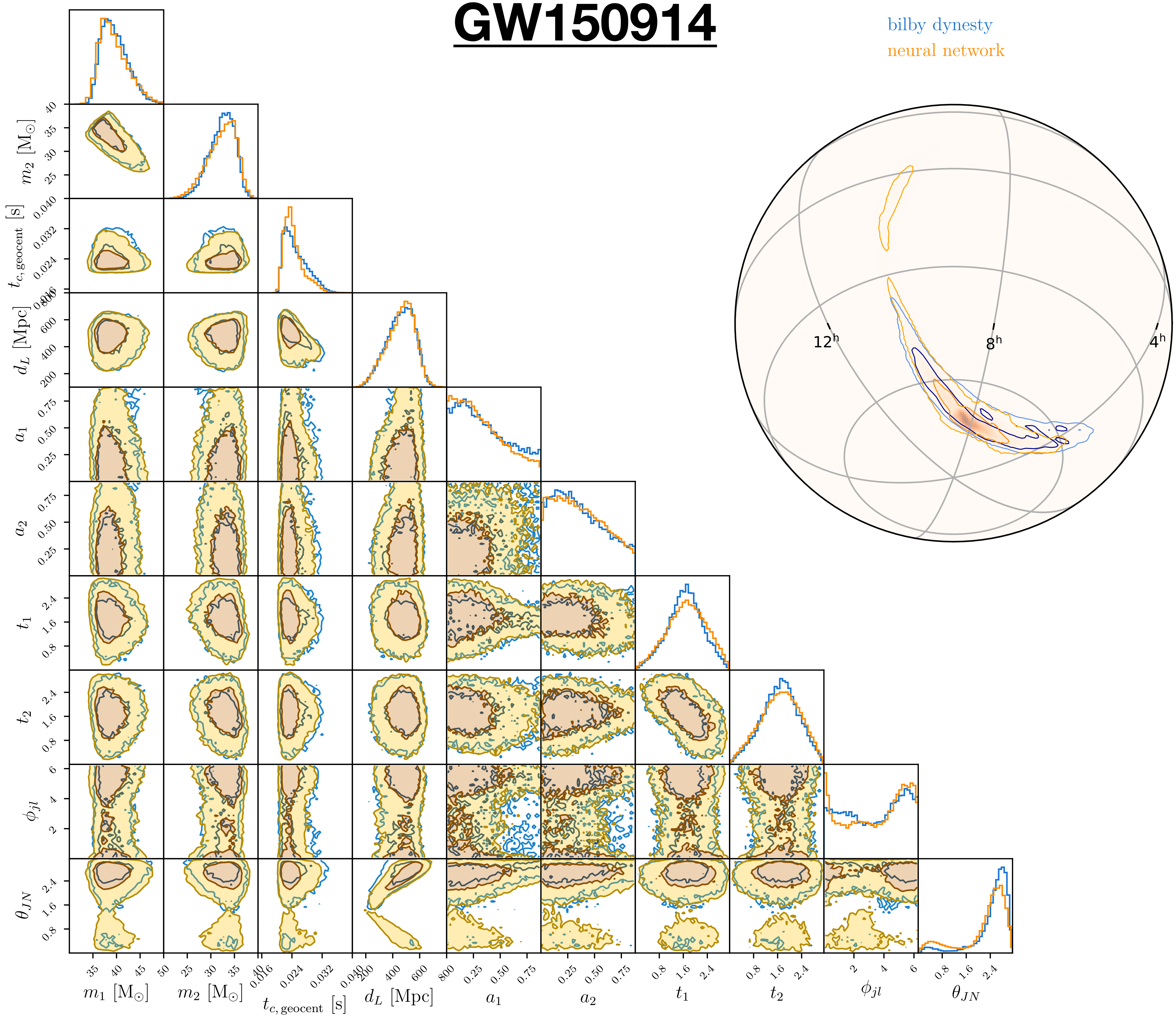
Application to binary black holes

- **Training**: 10^6-element training set; 500 epochs ~ few days



- **Inference**: Plug in strain for GW150914; thousands of samples / second

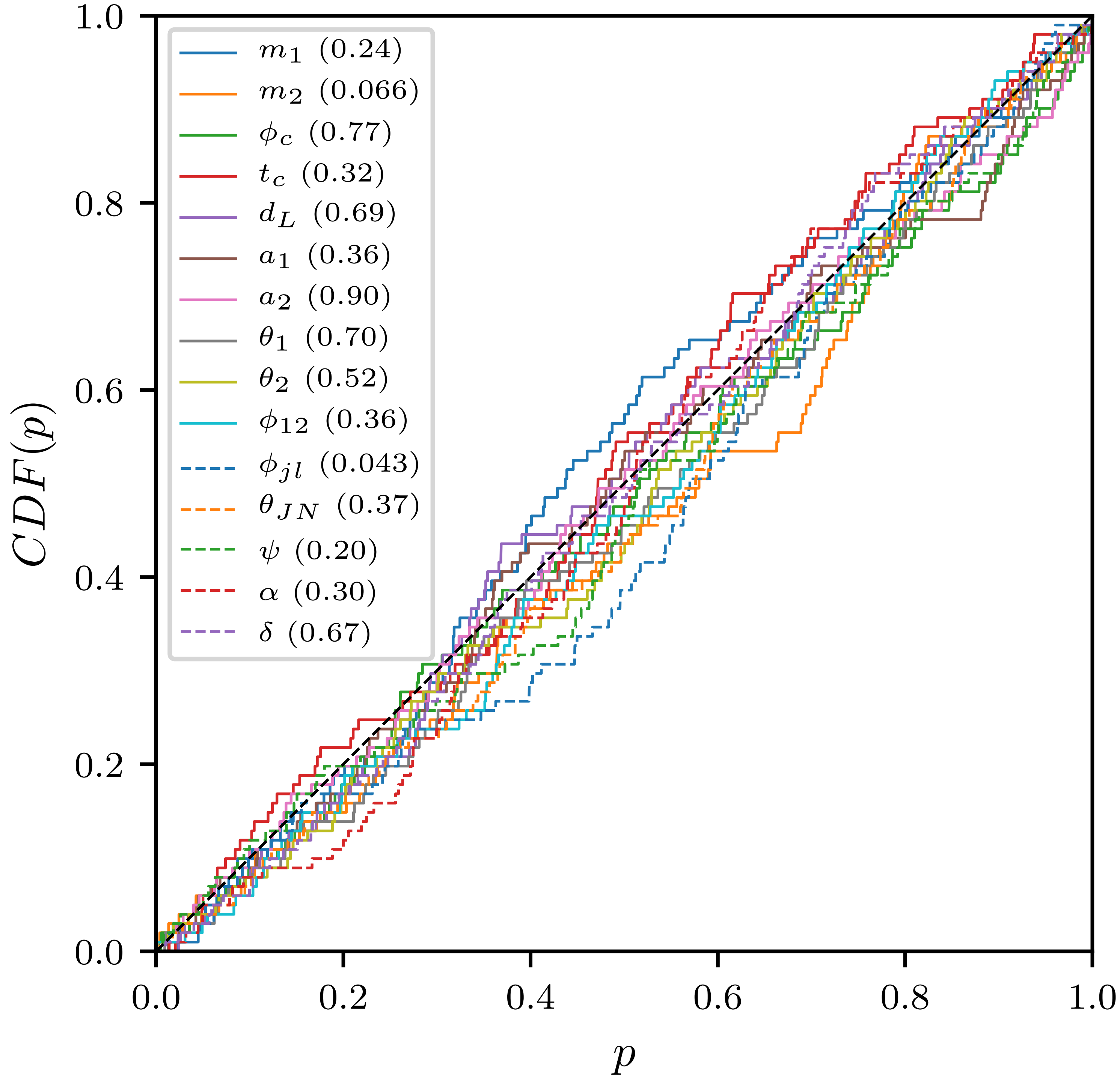
GW150914



bilby dynesty
neural network
m_1 [M$_\odot$]
m_2 [M$_\odot$]
$t_{c,\,\mathrm{geocent}}$ [s]
d_L [Mpc]
a_1
a_2
t_1
t_2
ϕ_{jl}
θ_{JN}
12$^{\mathrm{h}}$
8$^{\mathrm{h}}$
4$^{\mathrm{h}}$

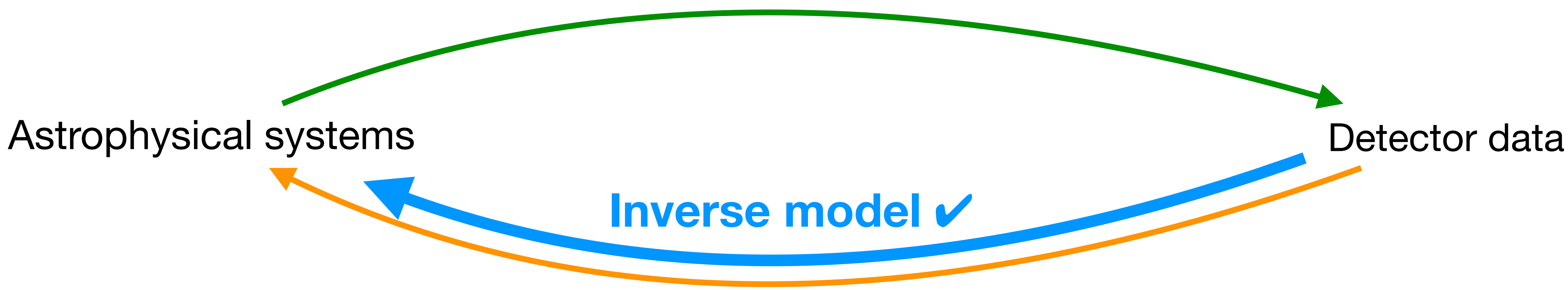
P—P plot

- We have built posterior model for any $s \sim p(s)$.
- Perform inference on 100 injections. (A few minutes total.)
- For each 1D marginalized posterior, plot CDF of percentile values of true parameters.



Summary

- Using simulation-based inference and normalizing flows, can build **non-iterative inverse models** for system parameters given the data.



Fast direct sampling for any $s \sim p(s)$ used for training.

- Performed accurate parameter estimation on GW150914 strain data in full 15D space.
- Next: Improve treatment of detector noise to allow variation from event to event, fully amortizing training time over inference runs.
- Code available: https://github.com/stephengreen/lfi-gw

Outlook

- In addition to fast inference, normalizing flows and simulation-based inference can give more accurate inference than standard methods because an explicit likelihood function is not required!

- Many potential applications for gravitational waves:

1. Population inference (see work of K. Wong *et al*).

2. Move beyond the idealization of stationary Gaussian noise, reducing systematic error present in standard analyses. Learn to remove glitches.

3. Extend to long complicated signals, like binary neutron stars and extreme mass-ratio inspirals for LISA.

4. Expand the parameter space to multiple simultaneous events, as predicted for LISA.

THANK YOU