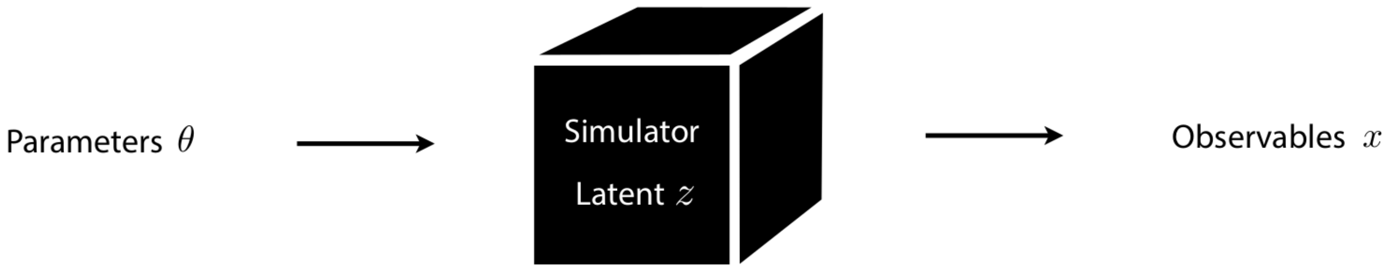
Highlights

PHYSTAT Statistics meets Machine Learning
September 12, 2024

Gilles Louppe
g.louppe@uliege.be



Prediction:
- Well-motivated mechanistic, causal model
- Simulator can generate samples $x \sim p(x|\theta)$

Inference:
- Interactions between low-level components lead to challenging inverse problems
- Likelihood $p(x|\theta) = \int \mathrm{d}z \; p(x, z|\theta)$ is intractable

Takeaway 0: Simulation-based inference is taking off!

Simulation-Based Inference in PHY-STAT

11:45	Simulation-based machine learning for gravitational-wave analysis	45m
17:10	Anomaly aware machine learning for dark matter direct detection at the DARWIN experiment	25m
14:25	lsbi: linear simulation based inference	25m
11:15	pop-cosmos: an interpretable generative model for the galaxy population over cosmic time	30m
17:35	Feldman-Cousins' ML Cousin	25m
11:00	Cosmology and machine learning	45m
18:09	Advanced techniques for Simulation Based Inference in collider physics	1m
18:10	SBI for wide field weak lensing	1m
18:13	Accounting for Selection Effects in Supernova Cosmology with Simulation-Based Inference and Hierarchical Bayesian Modelling	1m
11:00	Simulation-based Inference (SBI)	45m

Workshop in Munich this summer

This workshop alone !

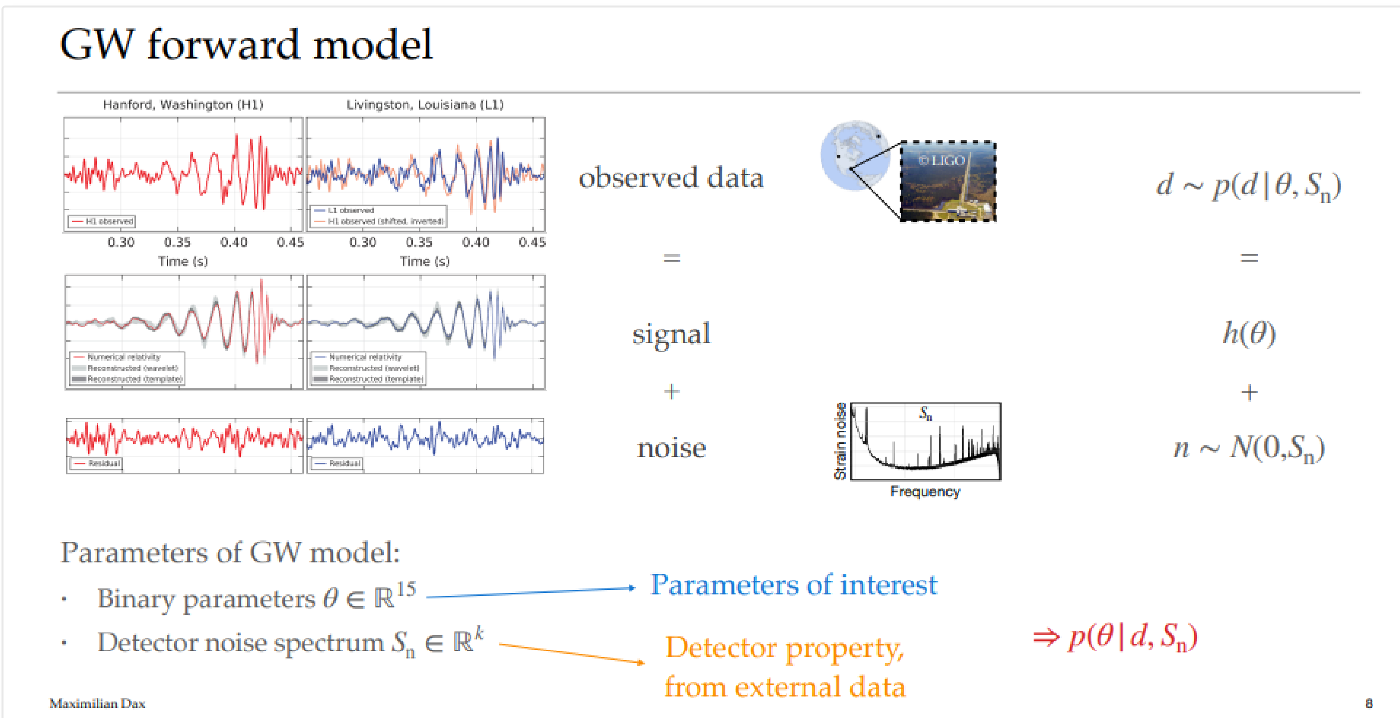
4

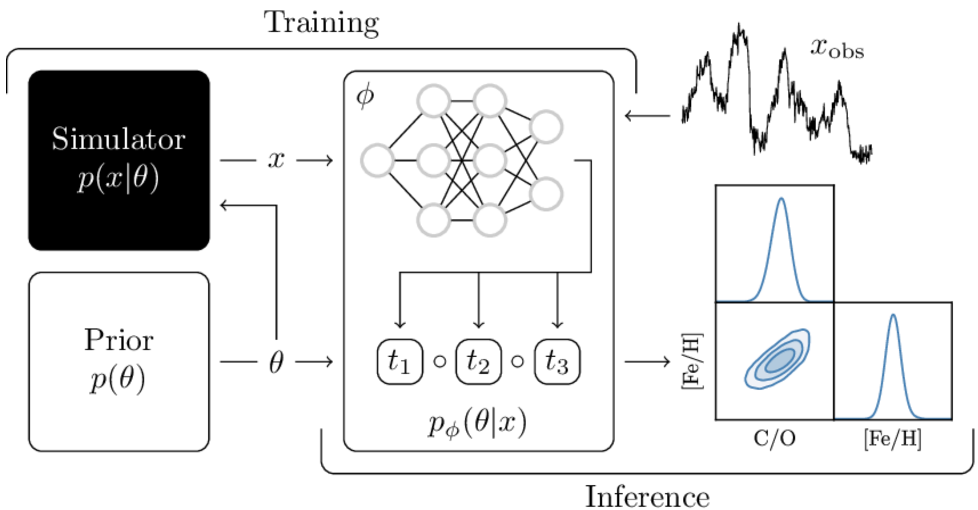
Credits: Aihsik Ghosh.

Takeaway 1: SBI is also valuable when likelihoods are tractable!

For example, with deterministic simulators and additive noise,

$$p(x|\theta) = \mathcal{N}(x; f(\theta), \Sigma(\theta)).$$





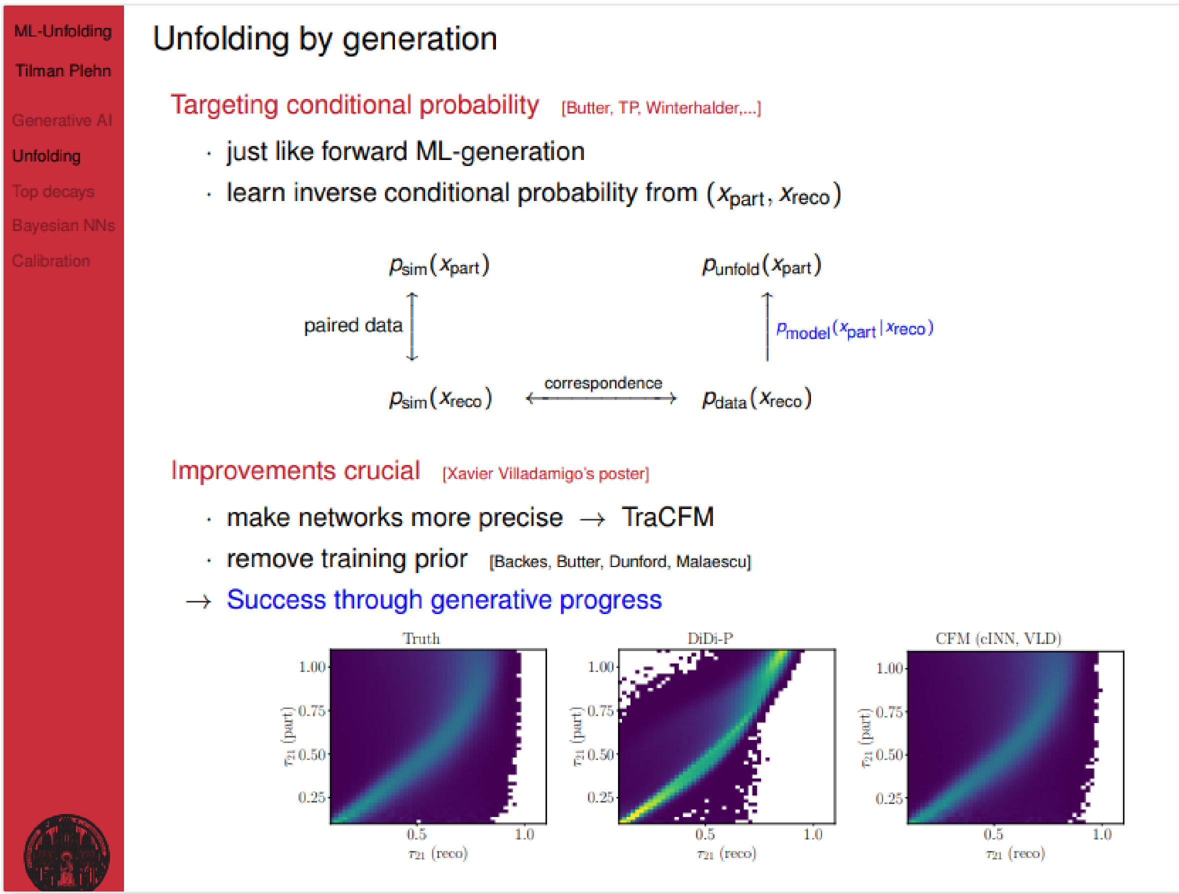
To account for measurement noise and make the simulation model similar to instrumental data, we consider a Gaussian noise model with a standard deviation σ. The spectra $f(\theta)$ generated by `petitRADTRANS` are randomly perturbed with additive noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$, where $\epsilon \in \mathbb{R}^{379}$ is a vector of random noise instances in each wavelength bin. Here we assume the same noise variance in each wavelength bin for the sake of simplicity, but more complex noise models (including noise covariance) could be used in our simulator. The final simulator output is given by $x = f(\theta) + \epsilon$.

—

Credits: Vasist et al, 2023.

Takeaway 2: Unfolding = Aggregated amortized posteriors.

$$p(x_{\text{part}}) = \int p(x_{\text{reco}})p(x_{\text{part}}|x_{\text{reco}})dx_{\text{reco}}$$



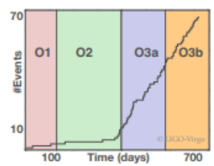
Credits: Tilman Plehn.

Takeaway 3: Requirements for accuracy and reliability are strict.

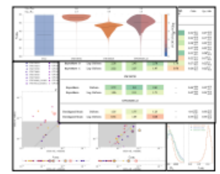
Motivation

Why GWs need ML

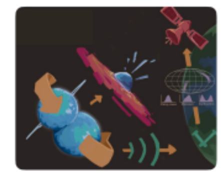
Increasing event rate



Large-scale analyses



Follow-up searches



Why ML needs GWs

Strict requirements for **accuracy**, **reliability** and **interpretability**

Complexity of GW data

$\Rightarrow$ GW data analysis pushes existing ML past its limits

Maximilian Dax 2

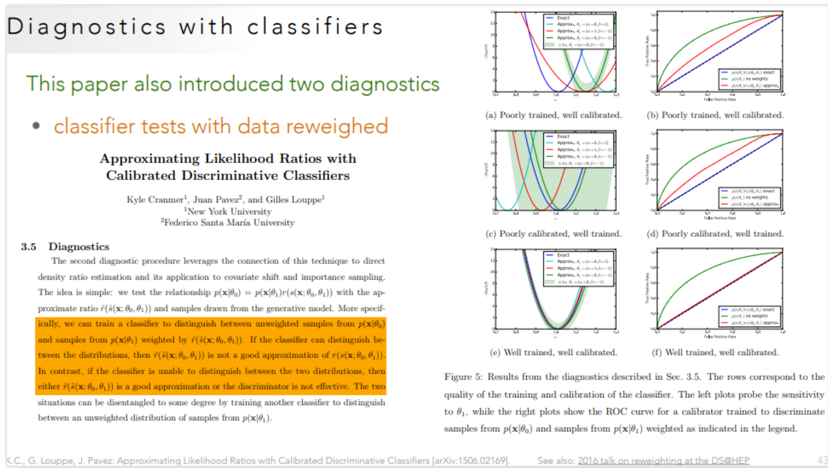
Challenges for NSBI:

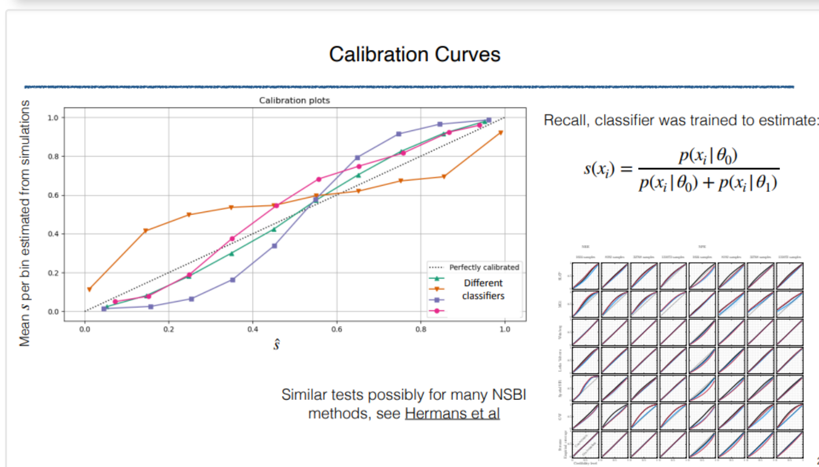
- Robustness: Design and validation
- Uncertainties: Quantifying and propagating systematics
- Neyman Construction: Throwing toys in high-dimensions

—

Credits: Maximilian Dax, Aishik Ghosh.

Takeaway 4: Use diagnostics* to assess the quality of the approximation.

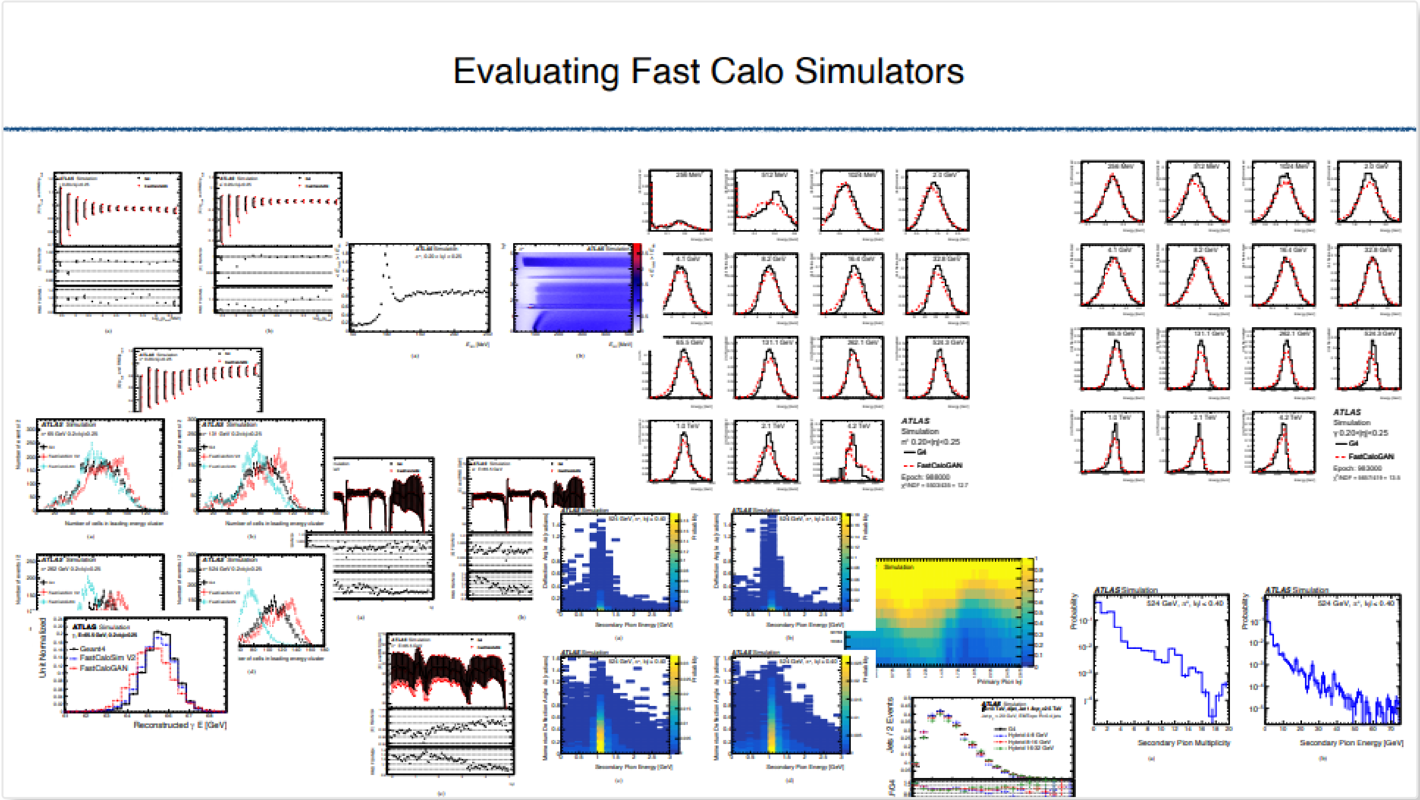




—

*None of those diagnostics provide sufficient guarantees.

Credits: Kyle Cranmer, Aishik Ghosh.



Takeaway 4b: ... and look at (too) many plots!

What if diagnostics fail?

Takeaway 5: More data, more parameters, more compute.

i.e., $S_n^{(i)} \sim p(S_n)$, $n^{(i)} \sim p(S_n^{(i)})$. We construct training sets based on 5×10^6 sets of intrinsic parameters, but by sampling extrinsic parameters and noise realizations during training, the effective size of the training set is infinite in these dimensions.

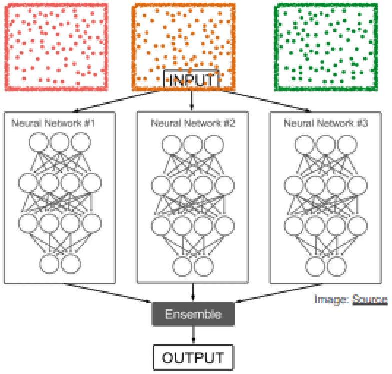
We also increase the total number of coupling transforms to 30 from 15 in [36]. The flow therefore consists of 300 hidden layers. In total, the embedding network and the flow combined have $1.31 \cdot 10^8$ learnable parameters for $n_{\text{detectors}} = 2$ and $1.42 \cdot 10^8$ for $n_{\text{detectors}} = 3$.

for validation to check for overfitting. Since the training and validation loss are in close agreement in Fig. 5 we conclude that overfitting is minimal. Training 450 epochs with a batch size of 4096 takes roughly 10 days on a single NVIDIA A100 GPU.[3]

Takeaway 6: Regularize the inference network.

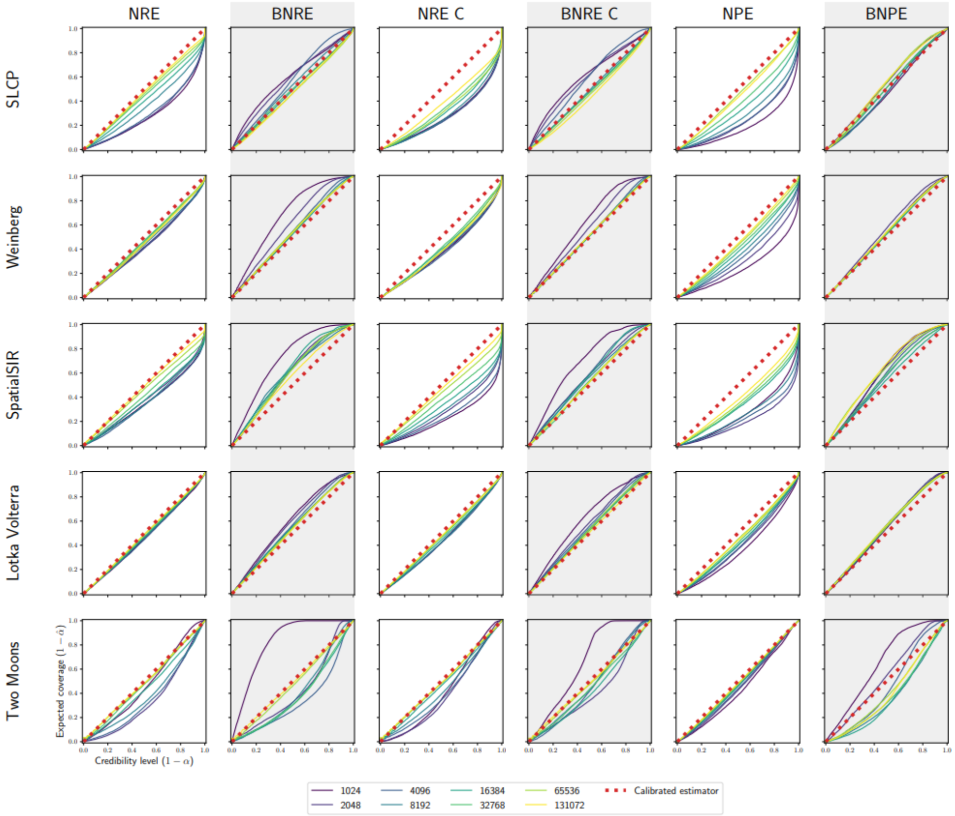
Quantifying uncertainty on estimated likelihood..

- Train an ensemble of networks, **each on a bootstrapped version of the training dataset**
  - Or Bayesian networks ? [Delaunoy et al, arXiv2408.15136]
- The spread in their prediction provides the uncertainty due to limited training statistics, and random network initialisation
- **Ensemble average used as final prediction**, so what's the uncertainty on that ?
  - Too expensive to train thousands of ensembles
  - Create bootstrapped ensembles ?
    - Each network trained on bootstrapped training dataset ?



Ensembles and Bayesian model averaging (BNNs) smooth out the approximation.

Credits: Aishik Ghosh



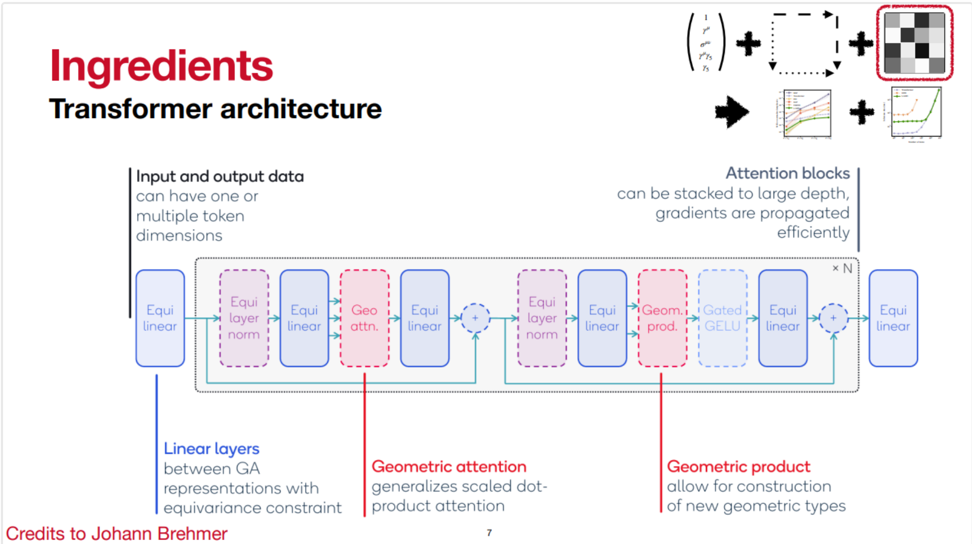
Approximations can also be regularized to be conservative (BNRE, BNPE).

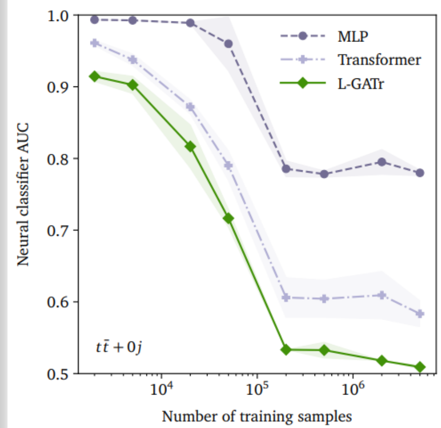
—

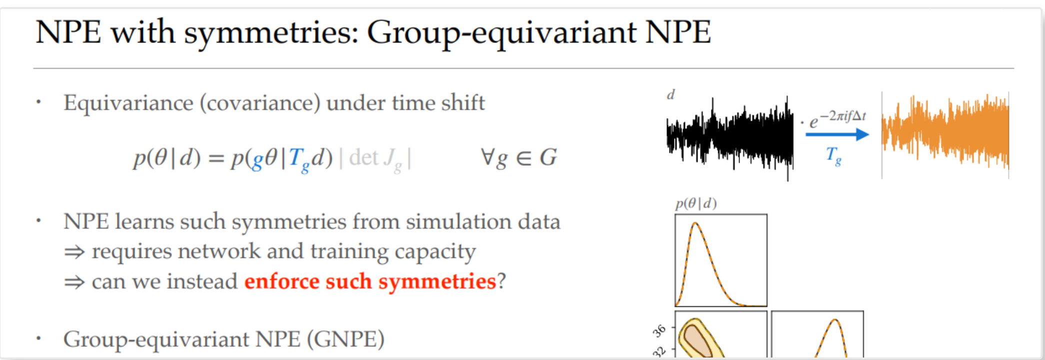
Credits: Delaunoy et al, 2022; Delaunoy et al, 2023.

Not enough!

Takeaway 7: Hardcode domain knowledge in the inference network.







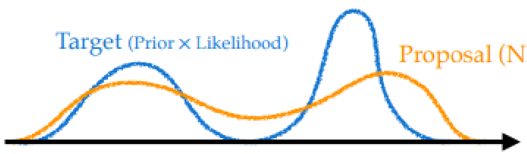
—
Credits: Jonas Spinner, Maximilian Dax.

Takeaway 8: Importance sampling is a cheat code for asymptotically exact inference with imperfect inference networks.

Dax+ (PRL 2023)

Importance-sampled NPE (NPE-IS)

- If likelihood is tractable, can reweight NPE results



$$\theta_i \sim q(\theta \mid d)$$

$$w_i = \frac{p(\theta_i)p(d \mid \theta_i)}{q(\theta_i \mid d)}$$

⇒ **asymptotically exact**

- **Effective number of samples** as performance metric

$$n_{\text{eff}} = \left(\Sigma_i w_i\right)^2 / \Sigma_i \left(w_i^2\right) \qquad \epsilon = n_{\text{eff}}/n \in (0,1]$$

⇒ **verification** without for ground truth posterior

- Estimate of **Bayesian evidence**

$$p(d) = \frac{1}{n}\sum_i w_i \qquad \sigma_{\log p(d)} = \sqrt{(1-\epsilon)/(n \cdot \epsilon)}$$

⇒ unbiased & precise estimate of **evidence**

Maximilian Dax 14

—
Credits: Maximilian Dax.

If $p(x_i|\theta_i)$ is intractable, the correction factor w_i can be estimated by a classifier.

- Classifier-based diagnostics provide diagnostics, but also a way to correct the approximation.
- If repeated, then one obtains a **sequential** NRE algorithm.

Takeaway 9: Nuisance parameters are a nuisance.

Curse of dimensionality for nuisance parameters

The traditional binned-template analysis approach uses a fixed interpolation / “template morphing” strategy

- Dependence on the parameters of interest are usually very well motivated
- makes assumptions about factorization of systematics that might not be true
- … either way, fixed parametric form makes it VERY sample efficient

In contrast, parametrized NN is physics-agnostic and the interpolation is non-parametric

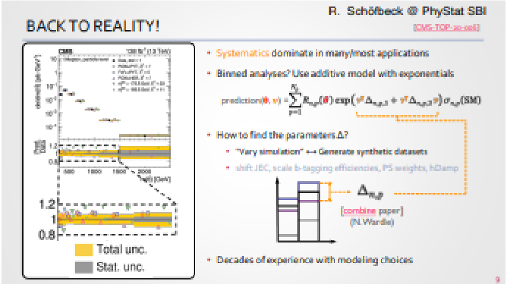
- Flexible, but requires many samples for a high-dimensional nuisance parameter space
- Curse of dimensionality

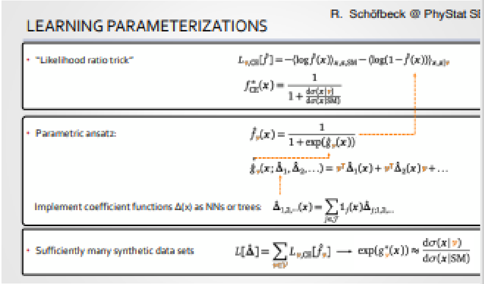
Is there a way to apply similar assumptions as template-based morphing strategy in neural SBI context?

—
Credits: Kyle Cranmer.

Recent developments shown at PhyStat SBI

R. Schöfbeck & Jay Sandesara both showed work in that direction at PhyStat SBI





- **Basic idea:** use same interpolation point wise in x, replace histogram bin with function of x.

where for each component $d\sigma_p(\boldsymbol{x}|\boldsymbol{\theta},\boldsymbol{\nu})$, we can obtain event samples from Eq. 23, Eq. 30, or Eq. 31. Next, we factorize the systematic effects and the POI dependence. The SM point corresponds to $\boldsymbol{\theta} = \boldsymbol{\nu} = \boldsymbol{0}$ and for each $d\sigma_p(\boldsymbol{x}|\boldsymbol{\theta},\boldsymbol{\nu})$ we have

$$\frac{d\sigma_p(\boldsymbol{x}|\boldsymbol{\theta},\boldsymbol{\nu})}{d\sigma_p(\boldsymbol{x}|\boldsymbol{0},\boldsymbol{0})} = \frac{d\sigma_p(\boldsymbol{x}|\boldsymbol{\theta},\boldsymbol{\nu})}{d\sigma_p(\boldsymbol{x}|\boldsymbol{0},\boldsymbol{\nu})}\frac{d\sigma_p(\boldsymbol{x}|\boldsymbol{0},\boldsymbol{\nu})}{d\sigma_p(\boldsymbol{x}|\boldsymbol{0},\boldsymbol{0})} \approx \frac{d\sigma_p(\boldsymbol{x}|\boldsymbol{\theta},\boldsymbol{0})}{d\sigma_p(\boldsymbol{x}|\boldsymbol{0},\boldsymbol{0})}\frac{d\sigma_p(\boldsymbol{x}|\boldsymbol{0},\boldsymbol{\nu})}{d\sigma_p(\boldsymbol{x}|\boldsymbol{0},\boldsymbol{0})} = \underbrace{\frac{d\sigma_p(\boldsymbol{x}|\boldsymbol{\theta},\boldsymbol{0})}{d\sigma_p(\boldsymbol{x}|\mathrm{SM})}}_{R_p(\boldsymbol{x}|\boldsymbol{\theta})}\underbrace{\frac{d\sigma_p(\boldsymbol{x}|\boldsymbol{0},\boldsymbol{\nu})}{d\sigma_p(\boldsymbol{x}|\mathrm{SM})}}_{S_p(\boldsymbol{x}|\boldsymbol{\nu})}. \quad (58)$$

The approximation is valid if the relative SMEFT effects are independent of the relative systematic effects, i.e.,

$$\frac{d\sigma_p(\boldsymbol{x}|\boldsymbol{\theta},\boldsymbol{\nu})}{d\sigma_p(\boldsymbol{x}|\boldsymbol{0},\boldsymbol{\nu})} \approx \frac{d\sigma_p(\boldsymbol{x}|\boldsymbol{\theta},\boldsymbol{0})}{d\sigma_p(\boldsymbol{x}|\boldsymbol{0},\boldsymbol{0})}. \quad (59)$$

The factor

$$R_p(\boldsymbol{x}|\boldsymbol{\theta}) \simeq \frac{d\sigma_p(\boldsymbol{x}|\boldsymbol{\theta},\boldsymbol{0})}{d\sigma_p(\boldsymbol{x}|\mathrm{SM})} \quad (60)$$

approximates the SMEFT variations and is a polynomial in $\boldsymbol{\theta}$. It can be obtained from one of the techniques in Refs. [8–17]. Systematic effects are parametrized by

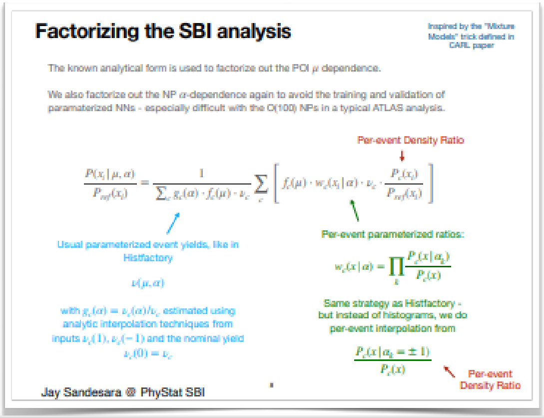
$$S_p(\boldsymbol{x}|\boldsymbol{\nu}) \simeq \frac{d\sigma_p(\boldsymbol{x}|\boldsymbol{0},\boldsymbol{\nu})}{d\sigma_p(\boldsymbol{x}|\mathrm{SM})}. \quad (61)$$

holds to good accuracy. We estimate $\hat{S}_p(\boldsymbol{x}|\boldsymbol{\nu})$ with Eq. 53 for each process. Following the same steps, we can furthermore factorize $\hat{S}_p(\boldsymbol{x}|\boldsymbol{\nu})$ into mutually uncorrelated groups of systematic uncertainties and train each factor individually. For example, uncorrelated one-parameter systematic uncertainties at quadratic accuracy reduce the surrogate to

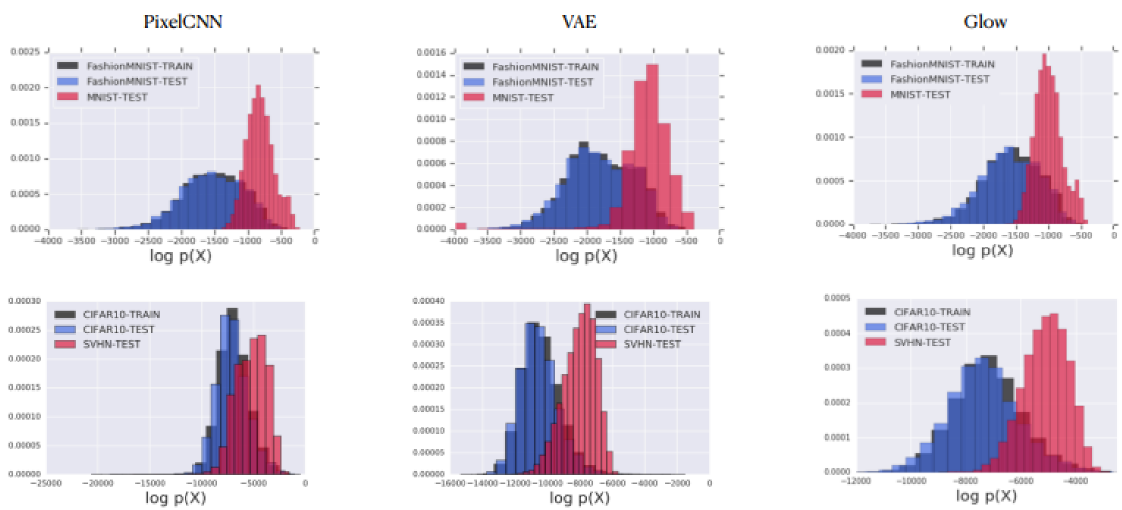
$$\hat{S}_p(\boldsymbol{x}|\boldsymbol{\nu}) = \prod_{k=1}^{K} \exp\left(\nu_k \hat{\Delta}_{p,k,1}(\boldsymbol{x}) + \nu_k^2 \hat{\Delta}_{p,k,2}(\boldsymbol{x})\right) \quad (63)$$

with $2K$ real-valued functions $\hat{\Delta}_{p,k,1}(\boldsymbol{x})$ and $\hat{\Delta}_{p,k,2}(\boldsymbol{x})$ for each p. In most cases, first or second-degree polynomials provide excellent approximations although this is not a limitation of our methodology.

https://arxiv.org/abs/2406.19076



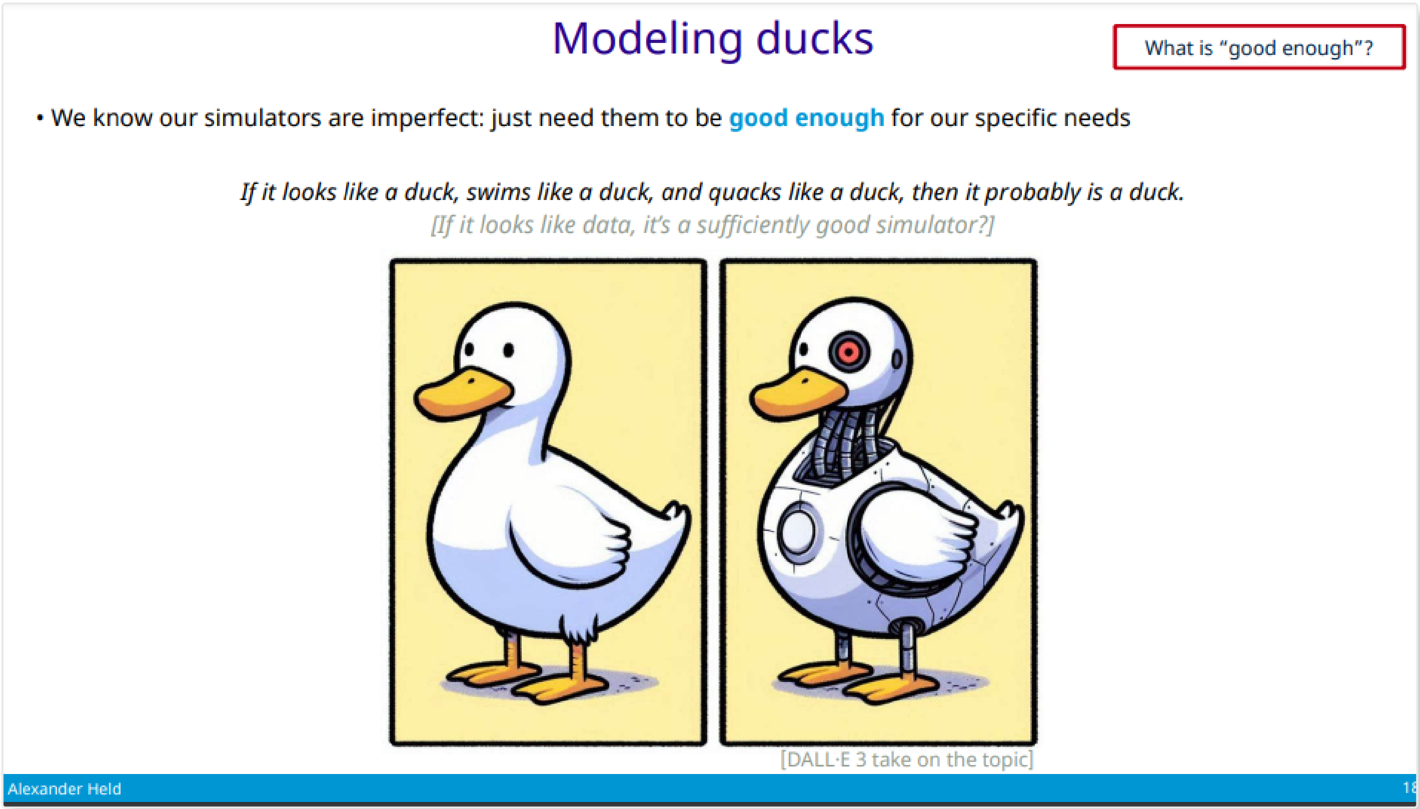
Takeaway 10: Neural density estimators do not know what they do not know.



6

Eric Nalisnick, Akihro Matsukawa, Yee Whye Teh, Dilan Gorur, Balaji Lakshiminarayan. "Do Deep Generative Models Know What They Don't Know?" *ICLR 2019*.

—
Credits: Lily H. Zhang.



Simulators are imperfect, but good enough?

Wait a minute... If the simulator is misspecified,
then data data may be OOD for the inference network.

My takeaways

1. Simulation-based inference is also valuable when likelihoods are tractable!
2. Unfolding = Aggregated amortized posteriors.
3. Requirements for accuracy and reliability are strict.
4. Use diagnostics to assess the quality of the approximation.
5. More data, more parameters, more compute.
6. Regularize the inference network.
7. Hardcode domain knowledge in the inference network.
8. Importance sampling is a cheat code for asymptotically exact inference with imperfect inference networks.
9. Nuisance parameters are a nuisance.
10. Neural density estimators do not know what they do not know.