Basics of simulation-based inference

Noemi Anau Montel

MPA Postdoc/Staff Lecture Series on Cosmology
27th November 2025

Contents

1 Introduction

Science involves connecting theoretical models to empirical observations. This connection can be viewed in two complementary directions: the *forward problem* of predicting data $\boldsymbol{x}$ from model parameters Θ, and the *inverse problem* of inferring parameters Θ from observed data $\boldsymbol{x}_0$. While the forward problem is conceptually well-defined through the physical or computational model, the inverse problem is more challenging as it requires navigating the space of all possible parameter values that could explain the observations. This forms the basis of statistical inference.

1.1 The Bayesian Framework

Bayesian inference provides a principled framework for solving inverse problems by treating parameters as random variables and quantifying uncertainty through probability distributions. Consider a model characterized by parameters $\Theta \in \mathbb{R}^{D_\Theta}$ that generates observable data $\boldsymbol{x} \in \mathbb{R}^{D_x}$ according to a data-generating process (or likelihood) $p(\boldsymbol{x}|\Theta)$. Given an observation $\boldsymbol{x}$, Bayesian inference aims to compute the posterior distribution:

$$p(\Theta|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\Theta)p(\Theta)}{p(\boldsymbol{x})}, \tag{1}$$

where $p(\Theta)$ encodes prior knowledge about the parameters and $p(\boldsymbol{x}) = \int p(\boldsymbol{x}|\Theta)p(\Theta)\mathrm{d}\Theta$ is the evidence or marginal likelihood, which normalizes the posterior.

Figure 1 illustrates the relationships between these quantities. The joint distribution $p(\boldsymbol{x}, \Theta) = p(\boldsymbol{x}|\Theta)p(\Theta)$ (center panel) describes the probabilistic relationship between parameters and data. Marginalizing over Θ yields the evidence $p(\boldsymbol{x})$ (top panel), while the likelihood $p(\boldsymbol{x}|\Theta)$ (right panel) shows how data depend on parameters. The prior $p(\Theta)$(left panel) encodes initial beliefs, and the posterior $p(\Theta|\boldsymbol{x})$ (bottom panel) represents updated beliefs after observing data.

While Bayes' theorem provides an elegant solution to the inverse problem, its practical implementation is complex. In the toy example illustrated in Figure 1, all quantities (prior, likelihood, joint, evidence, and posterior) can be computed analytically or through simple numerical integration. However, for realistic scientific problems involving complex models or high-dimensional parameter spaces, direct computation of these distributions becomes intractable and computationally challenging.

1.2 Computational Approaches for Bayesian Inference

The posterior $p(\Theta|\boldsymbol{x})$ itself is the target of inference and typically has no closed form. Its shape, potentially multimodal, heavy-tailed, or concentrated in complex regions of parameter space, is unknown a priori and must be discovered through computation. Over decades of research, various computational strategies have been developed to address this challenge, each making different assumptions about what can be computed from the model. Table 1 summarizes the requirements for some of the main classes of inference algorithms.

Traditional sampling-based methods like Markov Chain Monte Carlo (MCMC) [1] and Nested Sampling (NS) [2] require evaluation of the likelihood $p(\boldsymbol{x}_0|\Theta)$ for a given observation $\boldsymbol{x}_0$. They construct sampling schemes that avoid explicit evidence integral computation by evaluating only likelihood ratios (MCMC) or by incrementally building the evidence as a byproduct of sampling from nested likelihood contours (NS). More sophisticated methods, like Hamiltonian Monte Carlo (HMC) [3] and Variational Inference (VI) [4], additionally require gradients $\nabla_\Theta \log p(\boldsymbol{x}_0|\Theta)$ to

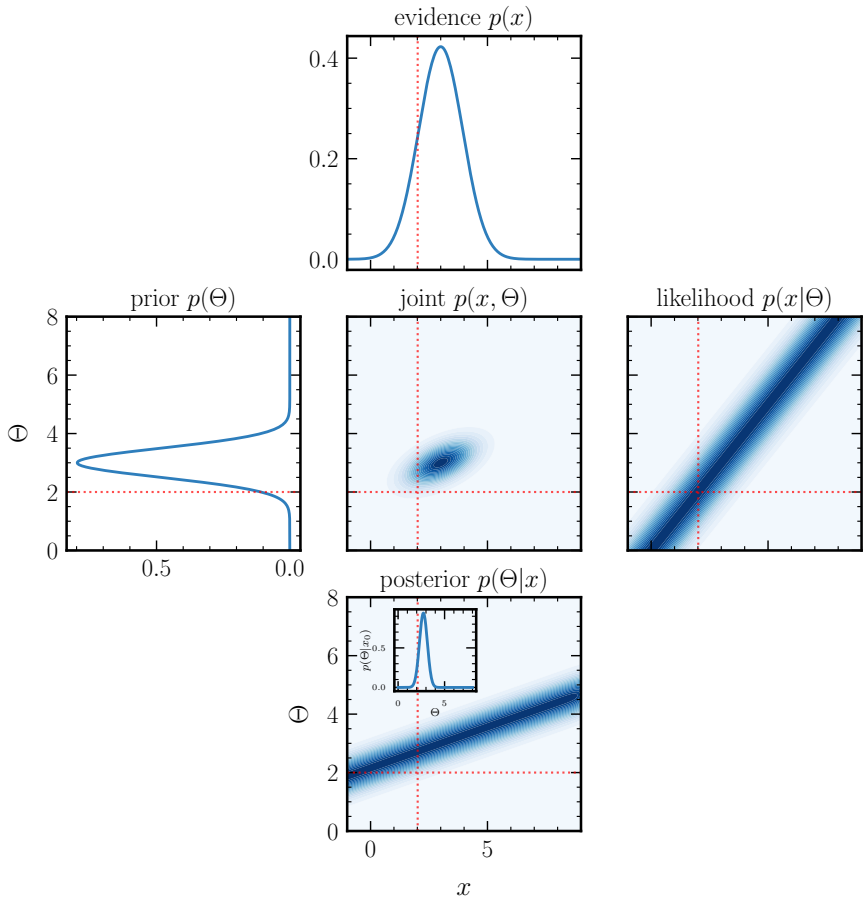

Figure 1: Relationships between probability distributions in Bayesian inference. The joint distribution $p(\boldsymbol{x}, \Theta) = p(\boldsymbol{x}|\Theta)p(\Theta)$ (*center*) connects all quantities in Bayes' theorem. Marginalizing over parameters yields the evidence $p(\boldsymbol{x})$ (*top*), while marginalizing over data yields the prior $p(\Theta)$ (*left*). Conditioning over data produces likelihood $p(\boldsymbol{x}|\Theta)$ (*right*), and over parameters produces the posterior $p(\Theta|\boldsymbol{x})$ (*bottom*). Red dashed lines mark a specific observation $\boldsymbol{x}_0$ generated from parameter Θ_0. The inset displays the posterior $p(\Theta|\boldsymbol{x}_0)$.

efficiently navigate high-dimensional parameter spaces, either through gradient-based dynamics (HMC) or optimization (VI).

Simulation-Based Inference (SBI) takes a different approach [5]. Rather than requiring likelihood evaluations, SBI methods need only the ability to generate joint samples $(\boldsymbol{x}, \Theta) \sim p(\boldsymbol{x}, \Theta) = p(\boldsymbol{x}|\Theta)p(\Theta)$ by sampling parameters from the prior and running the forward simulator. The terms *simulation-based inference*, *likelihood-free inference*, and *implicit likelihood inference* are used interchangeably in the literature to describe methods that perform Bayesian inference using only forward simulations. This makes SBI applicable to scenarios where traditional methods fail: when likelihoods are only implicitly defined through complex simulators, when the marginalization over latent variables is intractable, or when models are implemented as black-box codes without accessible likelihood functions. The trade-off is that SBI methods must learn surrogate models approximations of the posterior from simulated data.

	Joint Sampling $(\boldsymbol{x}, \Theta) \sim p(\boldsymbol{x}\|\Theta)p(\Theta)$	**Likelihood Evaluations** $p(\boldsymbol{x}_0\|\Theta) \in \mathbb{R}$	**Score Evaluations** $\nabla_\Theta \log p(\boldsymbol{x}_0\|\Theta) \in \mathbb{R}^{D_\Theta}$
MCMC	✗	✓	✗
NS	✗	✓	✗
HMC	✗	✓	✓
VI	✗	✓	✓
SBI	✓	✗	✗

Table 1: Requirements for different Bayesian inference methods based on the data model $p(\boldsymbol{x}|\Theta)$. MCMC and NS need a program that evaluates the data likelihood $p(\boldsymbol{x}|\Theta)$. In addition to this, HMC and VI need a program that can also evaluate $\nabla_\Theta \log p(\boldsymbol{x}|\Theta)$. Differently, SBI needs a stochastic program that outputs random samples $x, \Theta \sim p(\boldsymbol{x}|\Theta)p(\Theta)$.

This paradigm traces back to Donald Rubin's[1] work in the 1980s, where he advocated for computational methods that could handle models beyond those with analytically tractable likelihoods [6, Section 3.1]. An outline of what would become Approximate Bayesian Computation, the first widely-used SBI algorithm, already appears in Rubin's work.

2 Approximate Bayesian Computation

It is useful to start with the Approximate Bayesian Computation (ABC) algorithm, which is the classical analogue of neural network-based SBI. For more detailed reviews of ABC see Refs. [7, 8, 9]. ABC is a rejection sampling algorithm where, given a prior $\Theta \sim p(\Theta)$ proposed samples $\boldsymbol{x} \sim p(\boldsymbol{x} \mid \Theta)$ from the forward model are compared to the target observed data $\boldsymbol{x}_0$ with a hand-crafted distance measure based on some low-dimensional summary statistics $s(\boldsymbol{x})$, as for example

$$d(\boldsymbol{x}, \boldsymbol{x}_0) = \|s(\boldsymbol{x}) - s(\boldsymbol{x}_0)\| \, . \tag{2}$$

Samples from the approximate posterior are drawn with rejection sampling using an acceptance tolerance ϵ such that samples satisfy $d(\boldsymbol{x}, \boldsymbol{x}_0) < \epsilon$. Hence the posterior

$$p_{\mathrm{ABC}}(\Theta \mid \boldsymbol{x}) = \frac{\int_{d(\boldsymbol{x}_0,\boldsymbol{x})<\epsilon} \mathrm{d}\boldsymbol{x}\, p(\boldsymbol{x} \mid \Theta) p(\boldsymbol{\Theta})}{\int_{d(\boldsymbol{x}_0,\boldsymbol{x})<\epsilon} \mathrm{d}\boldsymbol{x}\, p(\boldsymbol{x})} \tag{3}$$

is guaranteed to converge to the true one for sufficiently informative summary statistics $s(\boldsymbol{x})$ and for $\epsilon \to 0$, i.e. $p_{\mathrm{ABC}}(\Theta \mid \boldsymbol{x}) \xrightarrow{\epsilon \to 0} p(\Theta \mid \boldsymbol{x})$. Whereas, when ϵ is non-zero, the approximate posterior is guaranteed to be broader than the true one, leading to conservative inference.

The algorithm is illustrated in a simple toy scenario in Fig. 2. The right panel shows the correlation of an informative summary statistic $s(\boldsymbol{x})$ of the simulations with the toy model parameter. The accepted points for observation $\boldsymbol{x}_0$ within the threshold ϵ are shown in red. The left panel shows the comparison between the true posterior and ABC's posterior approximated from the accepted points.

Over the years, ABC has been adopted in astrophysics and cosmology, accompanied by the development of specialized softwares, such as `CosmoABC` [10], `abcpmc` [11], and `astroABC` [12].

[1] Yes, the same Rubin from the Gelman-Rubin convergence diagnostic for Monte Carlo simulations.

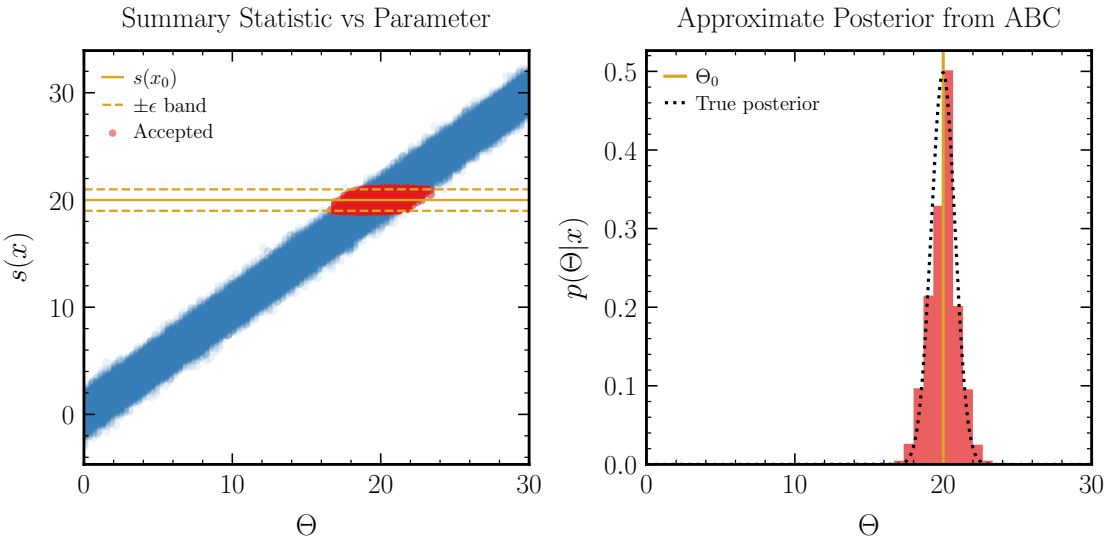

Figure 2: **Approximate Bayesian Computation example.** *Right:* Correlation of the informative summary statistic $s(\boldsymbol{x})$ of the simulations with the model parameter Θ, with the accepted point for observation $\boldsymbol{x}_0$ within the threshold ϵ in red. *Left:* Comparison between the true posterior and ABC's posterior approximated from the accepted points.

3 Neural Simulation-Based Inference

Despite their success and continuous progresses, there are some disadvantages of these classical SBI methods, that makes them unsuitable for high-dimensional and complex data. First, they rely on low-dimensional summary statistics $s(\boldsymbol{x})$ to compare simulations to the data, which may not retain all the information available in the data. Second, they suffer from the curse of dimensionality, with a required simulation budget that increases with the dimensionality of the parameter space [13].

These challenges are addressed in modern SBI algorithms thanks to advances in deep learning [14] and automatic differentiation [15]. On one hand, the development of neural network's architectures tailored to various data structures helps us process significantly more complex data. This allows for the optimization of learned data features w.r.t. some custom loss, instead of relying on hand-crafted summary statistics. On the other, neural network-based algorithms are being actively developed to estimate probability density distributions in high dimensions, overcoming the curse of dimensionality.

Moving beyond the classical SBI approach, in recent years, a number of new neural network-based SBI techniques have been proposed. We refer the reader to Ref. [5] for a general review, to Ref. [16] for benchmarks of the various algorithms, to Ref. [17] for a practical guide, and to Ref. [18] for a qualitative comparisons of SBI methods currently available as well as discussions about their pitfalls in the context of astrophysics and cosmology.[2] The widespread adoption of these methods in astrophysics and cosmology has also been driven by the development of specialized tools, such as, e.g., `pydelfi` [19], `sbi` [20], `swyft` [21], and `lampe` [22].

[2] An extensive list of works that use neural SBI in astrophysics, cosmology, and high energy physics can be found at https://github.com/smsharma/awesome-neural-sbi.

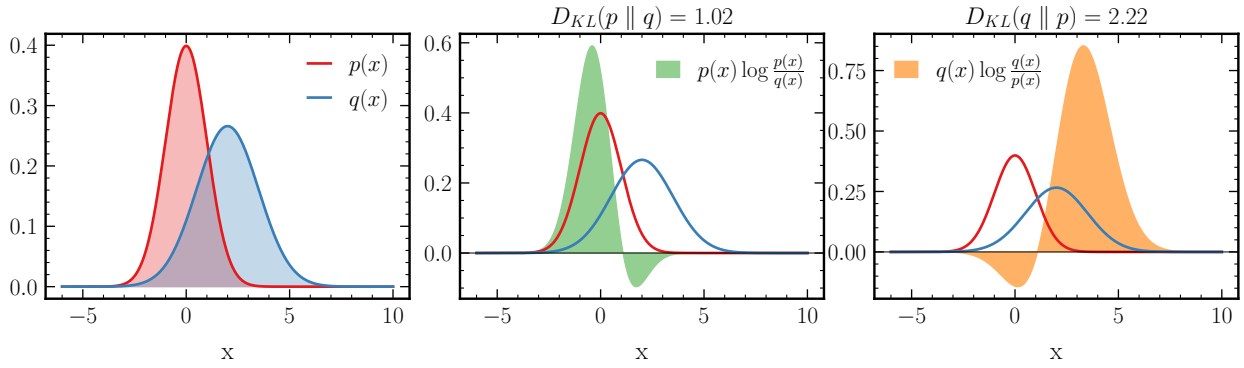

Figure 3: **Kullback-Leibler divergence.** *Right:* Two probability density distributions $p(x)$ and $q(x)$. *Center:* Forward KL divergence integrand. *Left:* Reverse KL divergence integrand.

In general, Bayes' theorem (Equation (1)) hints at a few different approaches for the implementation of neural SBI algorithms:

- Neural Posterior Estimation (NPE) which uses density estimation techniques to directly estimate the posterior $p(\Theta \mid \boldsymbol{x})$ [23, 24];
- Neural Posterior Estimation (NLE) which uses density estimation techniques to learn an approximation to the likelihood $p(\boldsymbol{x} \mid \Theta)$ [25, 26];
- Neural Posterior Estimation (NRE) which uses classifiers to approximate the likelihood-to-evidence or posterior-to-prior ratio $r(\Theta; \boldsymbol{x}) = \frac{p(\boldsymbol{x}|\Theta)}{p(\boldsymbol{x})} = \frac{p(\Theta|\boldsymbol{x})}{p(\Theta)}$ [27, 28, 29, 30].

3.1 Neural Posterior Estimation

NPE is the most straightforward method. It introduces a density estimator $q_\Phi^{\mathrm{NPE}}(\Theta \mid \boldsymbol{x})$, parametrized through a neural network with weights Φ (see Sec. 3.1.1 for more information about possible architectures), which is trained to approximate the posterior for parameters Θ given data $\boldsymbol{x}$,

$$q_\Phi^{\mathrm{NPE}}(\Theta \mid \boldsymbol{x}) \approx p(\Theta \mid \boldsymbol{x}) \; . \tag{4}$$

In order to train such density estimator $q_\Phi^{\mathrm{NPE}}(\Theta \mid \boldsymbol{x})$, we want to reduce the dissimilarity between it and the true posterior $p(\Theta \mid \boldsymbol{x})$.

A measure of dissimilarity of two probability distributions $p(x)$ and $q(x)$ is the **Kullback-Leibler (KL) divergence** [31], defined as

$$D_{\mathrm{KL}}(p(x) \| q(x)) = \int \mathrm{d}x \, p(x) \ln \left(\frac{p(x)}{q(x)} \right) . \tag{5}$$

The KL divergence, illustrated in Fig. 3, has three properties:

- $D_{\mathrm{KL}}(p\|q) \neq D_{\mathrm{KL}}(q\|p)$, i.e. is asymmetric;
- $p(x) = q(x) \quad \forall x \in \Omega \rightarrow D_{\mathrm{KL}}(p\|q) = D_{\mathrm{KL}}(q\|p) = 0$;
- $D_{\mathrm{KL}}(p\|q) \geq 0 \quad$ and $\quad D_{\mathrm{KL}}(q\|p) \geq 0 \quad \forall p \& q$.

To train an NPE density estimator, we minimize the *forward* KL divergence w.r.t. the weights Φ,

$$D_{\mathrm{KL}}(p\|q_\Phi^{\mathrm{NPE}}) = \int \mathrm{d}\Theta\, p\left(\Theta \mid \boldsymbol{x}\right) \ln\left(\frac{p\left(\Theta \mid \boldsymbol{x}\right)}{q_\Phi^{\mathrm{NPE}}\left(\Theta \mid \boldsymbol{x}\right)}\right) = \mathop{\mathbb{E}}_{\Theta\sim p(\Theta|\boldsymbol{x})}\left[\ln \frac{p\left(\Theta \mid \boldsymbol{x}\right)}{q_\Phi^{\mathrm{NPE}}\left(\Theta \mid \boldsymbol{x}\right)}\right] \tag{6}$$

As the expectation value is taken over samples from the true posterior distribution, we cannot evaluate the above quantity. To overcome this limitation, one averages over simulation data $\boldsymbol{x} \sim p(\boldsymbol{x})$.[3] The loss is now the expected forward KL divergence, where we have exploited the fact that $p(\Theta \mid \boldsymbol{x})p(\boldsymbol{x}) = p(\boldsymbol{x}, \Theta)$, which we know how to sample from;

$$\mathop{\mathbb{E}}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\left[D_{\mathrm{KL}}(p\|q_\Phi^{\mathrm{NPE}})\right] = \mathop{\mathbb{E}}_{\boldsymbol{x},\Theta\sim p(\boldsymbol{x},\Theta)}\left[\ln \frac{p(\Theta \mid \boldsymbol{x})}{q_\Phi^{\mathrm{NPE}}(\Theta \mid \boldsymbol{x})}\right] \tag{7}$$

$$= \underbrace{-\mathop{\mathbb{E}}_{\boldsymbol{x},\Theta\sim p(\boldsymbol{x},\Theta)} \ln q_\Phi^{\mathrm{NPE}}(\Theta \mid \boldsymbol{x})}_{\text{expected entropy}} + \underbrace{\mathop{\mathbb{E}}_{\boldsymbol{x},\Theta\sim p(\boldsymbol{x},\Theta)} \ln p(\Theta \mid \boldsymbol{x})}_{\text{constant w.r.t. } \Phi} . \tag{8}$$

The expected forward KL divergence can be decomposed in two terms. The latter one is constant w.r.t the weights Φ, so it will not contribute to the loss function. The first term is called "expected entropy" because it represents the expectation (average) of the log-probability $\ln q_\Phi^{\mathrm{NPE}}(\Theta \mid \boldsymbol{x})$ under the true joint distribution $p(\boldsymbol{x}, \Theta)$, which has the same mathematical form as the negative entropy $-H[q]$ but computed with respect to samples from p rather than q. It measures how well the approximate posterior q assigns probability mass to regions where the true distribution p actually places $(\boldsymbol{x}, \Theta)$ pairs.

NPE loss function is then given by the expected value over simulations of the negative log-probability estimator

$$\mathcal{L}^{\mathrm{NPE}}[\Phi] = -\mathop{\mathbb{E}}_{\boldsymbol{x},\Theta\sim p(\boldsymbol{x},\Theta)} \ln q_\Phi^{\mathrm{NPE}}(\Theta \mid \boldsymbol{x}) . \tag{9}$$

3.1.1 Density estimator

Applying NPE requires that the density estimator satisfies probability density properties, e.g. that it is normalized to one, i.e $\int \mathrm{d}\Theta\ q_\Phi^{\mathrm{NPE}}(\Theta \mid \boldsymbol{x}) = 1$. This normalization can be guaranteed by parameterizing q_Φ^{NPE} with architectures specifically designed for density estimation. For example, mixture density networks (MDNs) [32] are neural networks that output the parameters (means, variances, and mixture weights) of a Gaussian mixture model, which can represent multimodal conditional probability distributions. More recently, normalizing flows (NFs) [33, 34, 35] have become the preferred choice for NPE due to their greater expressiveness and tractable density evaluation.

Normalizing flows are a family of generative models with tractable distributions where both sampling and density evaluation can be efficient and exact. A normalizing flow is a transformation of a simple probability distribution (e.g., a standard normal) into a more complex distribution by a sequence of *invertible* and *differentiable* mappings. Formally, given a base distribution $z_0 \sim p(z_0)$, a normalizing flow is a series of forward mappings f_k with $k = 1, ..., K$, such that $z_k = f_k(z_{k-1})$. The new density can be sampled by sampling from the initial density and applying the chain of K transformations, i.e.

$$x = z_K = f_K \circ ... \circ f_1(z_0). \tag{10}$$

[3] Note that this is how losses are usually computed, formally with an expectation value over data, which becomes a Monte Carlo estimate over the available dataset.

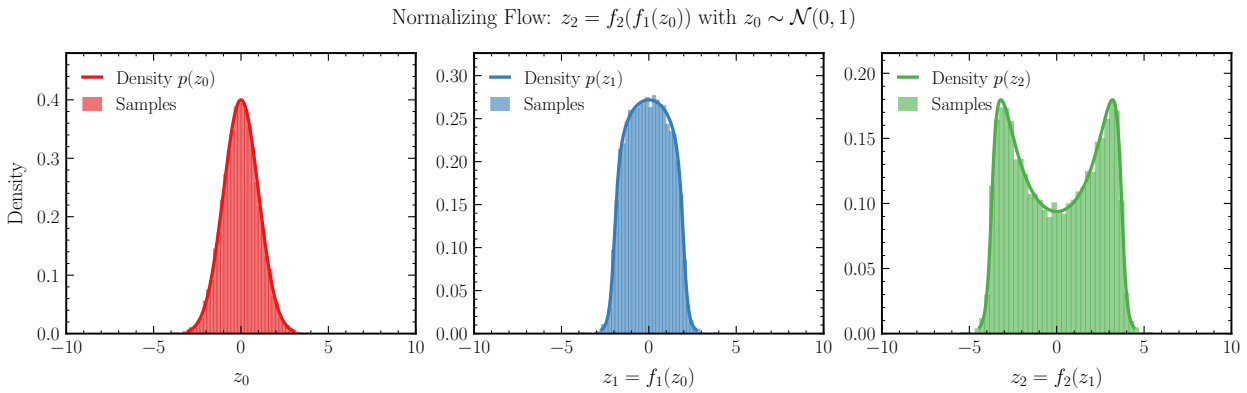

Figure 4: A **normalizing flow** is a transformation of a simple probability distribution (e.g., a standard normal, $p(z_0)$) into a more complex distribution by a sequence of invertible and differentiable mappings, here f_1 and f_2.

The density of a sample can be evaluated by transforming it back to the original simple distribution and then computing the product of i) the density of the inverse-transformed sample under this distribution and ii) the associated change in volume induced by the sequence of inverse transformations. The change in volume is the product of the absolute values of the determinants of the Jacobians for each transformation, as required by the change of variables formula. Putting everything together, the density of a sample under the new distribution can be computed as

$$\log p(x) = \log p(z_0) - \sum_{k=1}^{K} \log \left| \det J_{f_k}(z_{k-1}) \right| . \tag{11}$$

The result of this approach is a mechanism to construct new families of distributions by choosing an initial density and then chaining together some number of parameterized, invertible and differentiable transformations. An example of such transformations is given in Figure 4.

Normalizing flows can become powerful neural density estimators when parameterized by neural weights. In this setting, the trainable parameters Φ of the normalizing flow consist of both the parameters of the base distribution $p(z_0)$ and the parameters of all K transformations. These parameters can be optimized via gradient descent to minimize e.g. NPE loss in Equation (9), making normalizing flows suitable for learning complex conditional densities $q_\Phi^{\text{NPE}}(\Theta \mid \boldsymbol{x})$ in NPE.

Normalizing flow architectures have evolved to balance expressiveness with computational tractability. Autoregressive flows, such as Masked Autoregressive Flow (MAF) [36], decompose the transformation into a sequence of univariate conditional transformations, making the Jacobian triangular and thus allowing $\mathcal{O}(D)$ density evaluation (where D is the parameter space dimensionality), though sampling requires $\mathcal{O}(D^2)$ operations. Inverse Autoregressive Flow (IAF) [37] reverses this trade-off, enabling fast sampling but slower density evaluation. More recent architectures like Neural Spline Flows (NSF) [38] use monotonic splines as the transformation functions, increasing expressiveness through piecewise transformations. The choice of architecture depends on the specific requirements of the inference problem, with NSF currently being a popular choice in NPE applications due to their good empirical performance across diverse tasks.

3.1.2 Comparison with Variational Inference

It is interesting to compare NPE with its direct likelihood-based counterpart, VI (for a review see Ref. [39]). VI allows the approximation of extremely high-dimensional Bayesian posteriors with simple proposal distributions by solving an optimization problem. This is achieved by using a proposal distribution, the so-called variational posterior, $q_\Phi^{\mathrm{VI}}(\Theta \mid \boldsymbol{x}_0)$, to approximate the true posterior $p\left(\Theta \mid \boldsymbol{x}_0\right)$ and optimizing its parameters Φ using gradient descent [40]. VI's optimization objective is the *reverse* KL divergence

$$D_{\mathrm{KL}}(q_\Phi^{\mathrm{VI}} \| p) = \int \mathrm{d}\Theta \; q_\Phi^{\mathrm{VI}}\left(\Theta \mid \boldsymbol{x}_0\right) \ln\left(\frac{q_\Phi^{\mathrm{VI}}\left(\Theta \mid \boldsymbol{x}_0\right)}{p\left(\Theta \mid \boldsymbol{x}_0\right)}\right) = \mathop{\mathbb{E}}_{\Theta \sim q_\Phi^{\mathrm{VI}}(\Theta|\boldsymbol{x}_0)}\left[\ln \frac{q_\Phi^{\mathrm{VI}}\left(\Theta \mid \boldsymbol{x}_0\right)}{p\left(\Theta \mid \boldsymbol{x}_o\right)}\right] . \tag{12}$$

Similarly to the trick used for NPE, in order to avoid evaluating the inaccessible posterior $p\left(\Theta \mid \boldsymbol{x}_o\right)$, one exploits Bayes' theorem and substitute $p(\Theta|\boldsymbol{x}) = p(\boldsymbol{x}|\Theta)p(\Theta)/p(\boldsymbol{x})$, obtaining:

$$D_{\mathrm{KL}}(q_\Phi^{\mathrm{VI}} \| p) = \mathop{\mathbb{E}}_{\Theta \sim q_\Phi^{\mathrm{VI}}(\Theta|\boldsymbol{x}_o)}\left[\ln \frac{q_\Phi^{\mathrm{VI}}(\Theta \mid \boldsymbol{x}_o)}{p(\Theta \mid \boldsymbol{x}_o)}\right] \tag{13}$$

$$= \underbrace{\mathop{\mathbb{E}}_{\Theta \sim q_\Phi^{\mathrm{VI}}(\Theta|\boldsymbol{x}_o)}\left[\ln \frac{q_\Phi^{\mathrm{VI}}(\Theta \mid \boldsymbol{x}_o)}{p(\boldsymbol{x}_0 \mid \Theta)p(\Theta)}\right]}_{\equiv -\mathrm{ELBO}} + \underbrace{\ln p(\boldsymbol{x}_0)}_{\text{constant w.r.t. } \Phi} , \tag{14}$$

where we have introduced the evidence lower bound (ELBO). The term "evidence lower bound" reflects the fact that, since the KL divergence is non-negative, $D_{\mathrm{KL}}(q_\Phi^{\mathrm{VI}} \| p) \geq 0$, it follows that $\mathrm{ELBO} \leq \ln p(\boldsymbol{x}_0)$. Thus, maximizing the ELBO provides a lower bound on the log-evidence (or marginal likelihood) $\ln p(\boldsymbol{x}_0)$, with equality achieved when $q_\Phi^{\mathrm{VI}}(\Theta \mid \boldsymbol{x}_0) = p(\Theta \mid \boldsymbol{x}_0)$, i.e., when the variational posterior exactly matches the true posterior. Minimizing the reverse KL divergence is thus equivalent to maximizing the ELBO, and the VI loss is

$$\mathcal{L}^{\mathrm{VI}}[\Phi] = \mathrm{ELBO} \equiv \mathop{\mathbb{E}}_{\Theta \sim q_\Phi^{\mathrm{VI}}(\Theta|\boldsymbol{x}_0)}\left[\ln p\left(\boldsymbol{x}_0, \Theta\right) - \ln q_\Phi^{\mathrm{VI}}\left(\Theta \mid \boldsymbol{x}_0\right)\right] , \tag{15}$$

where the variational posterior $q_\Phi^{\mathrm{VI}}(\Theta \mid \boldsymbol{x}_0)$ must satisfy the same density estimation requirements as in NPE (i.e., it must be normalized and flexible enough to capture complex posterior distributions) and is therefore typically parameterized using normalizing flows or other expressive density estimators whose parameters Φ are optimized to minimize the loss.

The first thing to note here is that, unlike NPE, which trains an amortized inference network $q_\Phi^{\mathrm{NPE}}(\Theta \mid \boldsymbol{x})$ to handle any observation $\boldsymbol{x}$, VI optimizes the variational posterior $q_\Phi^{\mathrm{VI}}(\Theta \mid \boldsymbol{x}_0)$ specifically for a single observed dataset $\boldsymbol{x}_0$. This means that for each new observation, VI requires re-running the optimization procedure from scratch, whereas NPE can immediately evaluate the posterior for any new observation after a single training phase.[4]

Additionally, in NPE parameters are drawn from the model prior $p(\Theta)$ (see Equation (9)), whereas in VI they are drawn from the variational posterior $q_\Phi^{\mathrm{VI}}\left(\Theta \mid \boldsymbol{x}_0\right)$ (see Equation (15)). As a result, in the latter case, samples increasingly focus on regions with high data likelihood $p(\boldsymbol{x}_0 \mid \Theta)$, whereas in NPE there is no automatic focusing on regions with high data likelihood. Second, for VI the posterior $q_\Phi^{\mathrm{VI}}\left(\Theta \mid \boldsymbol{x}_0\right)$ must cover all parameters that the likelihood model is conditioned on, and omitting correlations (e.g. mean-field approximation) gives a lower bound on parameter

[4] This property is called "amortization" in machine learning.

uncertainties. On the other hand, in NPE, if we substitute $q_\Phi^{\mathrm{NPE}}(\Theta \mid \boldsymbol{x})$ with a low dimensional marginal, e.g. $q_\Phi^{\mathrm{NPE}}(\Theta_1 \mid \boldsymbol{x})$, we automatically get marginal posterior estimates, without performing integrals explicitly. Hence, omitting correlations generates correctly marginalized posteriors. The ability to directly perform marginal inference is shared among all other neural SBI algorithms, and it is one of the main differences with likelihood-based algorithms (see Section 5).

3.2 Neural Likelihood Estimation

Another choice for neural SBI is to perform NLE, where one trains an estimator for the likelihood probability density of the data $\boldsymbol{x}$ given some model parameters Θ,

$$q_\Phi^{\mathrm{NLE}}(\boldsymbol{x} \mid \Theta) \approx p(\boldsymbol{x} \mid \Theta) \,. \tag{16}$$

Following a similar reasoning as the one employed to derive NPE loss function, NLE loss function is given by the negative log-likelihood

$$\mathcal{L}^{\mathrm{NLE}}[\Phi] = - \mathop{\mathbb{E}}_{\boldsymbol{x},\Theta\sim p(\boldsymbol{x},\Theta)} \ln q_\Phi^{\mathrm{NLE}}(\boldsymbol{x} \mid \Theta) \,. \tag{17}$$

Again, specialized network architectures as normalizing flows are typically employed to guarantee that $q_\Phi^{\mathrm{NLE}}(\boldsymbol{x} \mid \Theta)$ is a properly normalized density function.

As a general guideline, the choice between NPE or NLE for different physical problems should consider the dimensionality of the data and parameter space, as well as the complexity of learning likelihood or posterior distribution's shape. Indeed, neural networks are easier to train when using high-dimensional input to yield low-dimensional output than vice versa. If the data is high-dimensional (e.g., images), then NPE may be a more suitable choice than NLE. Conversely, for high-dimensional posterior inference (i.e., when the dimension of the parameter space is large), NLE can outperform NPE, given good data summaries.

3.3 Neural Ratio Estimation

Bayes' theorem suggests a third estimable quantity: the likelihood-to-evidence or posterior-to-prior ratio

$$r(\Theta; \boldsymbol{x}) \equiv \frac{p(\boldsymbol{x} \mid \Theta)}{p(\boldsymbol{x})} = \frac{p(\Theta \mid \boldsymbol{x})}{p(\Theta)} = \frac{p(\boldsymbol{x}, \Theta)}{p(\boldsymbol{x})p(\Theta)} \,. \tag{18}$$

Ratio estimation rephrases Bayesian posterior inference as a binary classification problem. Given an implicitly defined model $\boldsymbol{x}, \Theta \sim p(\boldsymbol{x}, \Theta) = p(\boldsymbol{x} \mid \Theta)p(\Theta)$, the idea behind ratio estimation is to use a binary classifier to distinguish between data $\boldsymbol{x}$ and parameter Θ pairs drawn from two classes labeled by a binary variable C:

$$p(\boldsymbol{x}, \Theta \mid C = 1) = p(\boldsymbol{x}, \Theta) \qquad \text{and} \qquad p(\boldsymbol{x}, \Theta \mid C = 0) = p(\boldsymbol{x})p(\Theta) \,. \tag{19}$$

These two distributions correspond to drawing data and parameters jointly from the simulator, $\boldsymbol{x}, \Theta \sim p(\boldsymbol{x}, \Theta)$, or to drawing data and parameters marginally, $\boldsymbol{x}, \Theta \sim p(\boldsymbol{x})p(\Theta)$, by pairing unrelated parameters and data sampled independently (which can be obtained by shuffling the joint pairs).

Sampling $C = 0$ and $C = 1$ with equal probability, the decision function for the Bayes-optimal classifier [41] (i.e., the classifier that minimizes the Bayesian risk of misclassification) is:[5]

$$p(C = 1 \mid \boldsymbol{x}, \Theta) = \frac{p(\boldsymbol{x}, \Theta)}{p(\boldsymbol{x}, \Theta) + p(\boldsymbol{x})p(\Theta)} \equiv \sigma[\log r(\Theta; \boldsymbol{x})] \,, \tag{21}$$

where $\sigma(y) \equiv 1/(1 + e^{-y})$ is the sigmoid function. The equivalence in Equation (21), often called the "likelihood ratio trick" [27, 42], reveals that the Bayes-optimal classifier is directly related to the posterior-to-prior, likelihood-to-evidence, or joint-to-marginal ratio.

Thus, one can access the posterior-to-prior ratio by training a classifier to distinguish joint from marginal pairs (both easily obtainable from a forward simulator) and use it for inference. In practice, if the prior is tractable, $r(\Theta; \boldsymbol{x})$ provides direct access to the posterior density via $p(\Theta \mid \boldsymbol{x}) = r(\Theta; \boldsymbol{x})p(\Theta)$. Alternatively, one can use $r(\Theta; \boldsymbol{x})$ to weight prior samples, enabling posterior sampling even when the prior lacks a closed-form expression.

In NRE, the classifier in Equation (21) is parameterized by a neural network $d_\Phi(\boldsymbol{x}, \Theta)$ that takes a data-parameter pair as input and outputs an estimate of the ratio [6]

$$r(\Theta; \boldsymbol{x}) \approx \frac{d_\Phi(\boldsymbol{x}, \Theta)}{1 - d_\Phi(\boldsymbol{x}, \Theta)} \,. \tag{23}$$

The classifier architecture is typically a multilayer perceptron (MLP) or a residual neural network (ResNet)[43] when dealing with low-dimensional summary statistics. The network parameters Φ are optimized via stochastic gradient descent [40] to minimize the binary cross-entropy (BCE) loss [44]:

$$\mathcal{L}^{\text{NRE}}[\Phi] = -\int \mathrm{d}\boldsymbol{x}\mathrm{d}\Theta \left\{ p(\boldsymbol{x}, \Theta) \log d_{\boldsymbol{\Phi}}(\boldsymbol{x}, \Theta) \; + p(\boldsymbol{x})p(\Theta) \log \left[1 - d_{\boldsymbol{\Phi}}(\boldsymbol{x}, \Theta)\right] \right\} \,. \tag{24}$$

By training the neural network $d_\Phi(\boldsymbol{x}, \Theta)$ to estimate $r(\Theta; \boldsymbol{x})$ through this supervised classification task, we obtain an estimate of the posterior: $p(\Theta \mid \boldsymbol{x}) = r(\Theta; \boldsymbol{x})p(\Theta)$.

4 Data compression schemes

We have seen how neural SBI methods eliminate the need for hand-crafted acceptance thresholds ϵ through learned density estimation or classification. However, when data $\boldsymbol{x}$ are high-dimensional (images, spectra, cosmological fields), the challenge of mapping between high-dimensional data to low-dimensional parameters through summaries remains for density estimators and classifiers. Just

[5] Expanding the computations:

$$\begin{aligned} p(C = 1|\boldsymbol{x}, \Theta) &= \frac{p(\boldsymbol{x}, \Theta|C = 1)p(C = 1)}{p(\boldsymbol{x}, \Theta)} = \frac{p(\boldsymbol{x}, \Theta|C = 1)p(C = 1)}{p(\boldsymbol{x}, \Theta|C = 1)p(C = 1) + p(\boldsymbol{x}, \Theta|C = 0)p(C = 0)} \\ &= \frac{1}{1 + \dfrac{p(\boldsymbol{x}, \Theta|C = 0)p(C = 0)}{p(\boldsymbol{x}, \Theta|C = 1)p(C = 1)}} = \frac{1}{1 + \dfrac{p(\boldsymbol{x}, \Theta|C = 0)}{p(\boldsymbol{x}, \Theta|C = 1)}} = \frac{1}{1 + \dfrac{p(\boldsymbol{x})p(\Theta)}{p(x, \Theta)}} = \frac{1}{1 + \dfrac{1}{r(\Theta; \boldsymbol{x})}} \equiv \sigma(\log r(\Theta; \boldsymbol{x})) \end{aligned} \tag{20}$$

[6] This follows from the sigmoid function:

$$d_\Phi(\boldsymbol{x}, \Theta) = \sigma(\log r(\Theta; \boldsymbol{x})) = \frac{1}{1 + e^{-\log r(\Theta; \boldsymbol{x})}} = \frac{1}{1 + \frac{1}{r(\Theta; \boldsymbol{x})}} = \frac{r(\Theta; \boldsymbol{x})}{r(\Theta; \boldsymbol{x}) + 1} \tag{22}$$

.

as ABC required careful choice of summary statistics, neural SBI requires compression schemes to reduce the data dimensionality while preserving its information for parameter inference. In other words, a data compression from $\boldsymbol{x} \in \mathbb{R}^{D_x}$ to summary $s(\boldsymbol{x}) \in \mathbb{R}^{D_s}$ with $D_x > D_s$ should retain as much information about the parameters $\Theta \in \mathbb{R}^{D_\Theta}$ as possible, so that the posterior derived from the summary statistics approximates the true posterior:

$$p(\Theta|s(\boldsymbol{x}) \simeq p(\Theta|\boldsymbol{x}) \ . \tag{25}$$

We briefly review here three main approaches: hand-crafted summary statistics designed from domain knowledge, statistical compression methods that maximize Fisher information, and end-to-end neural compression trained jointly with the density estimator or classifier. The choice of compression scheme depends heavily on the specific application, the nature of the data, and available computational resources; the discussion below is not exhaustive but aims to provide a general understanding of the main strategies and their trade-offs.

4.1 Summary Statistics

The simplest approach is to apply domain-specific summary statistics before passing the compressed data to neural SBI algorithms. In some cases, domain knowledge identifies summary statistics that are provably sufficient or nearly optimal for inference, preserving all relevant information (known as sufficient statistics). However, for complex simulators, hand-crafted summaries may not be sufficient statistics and could discard important information. As a result, the trade-off is between the reliability and interpretability of well-understood summaries (even if not sufficient) versus the potential for capturing additional information through more flexible approaches.

4.2 Information-Theoretic Compression

For complex data, hybrid approaches have been developed that learn informative summary statistics from simulated data. For example, MOPED (Massively Optimized Parameter Estimation and Data compression) [45] performs optimal linear compression for Gaussian data with parameter-independent covariance, reducing D_x-dimensional data to $D_s = D_\Theta$ summaries (one per parameter) by projecting onto directions that maximize the Fisher information, requiring numerical derivatives of the data mean at a fiducial point. Score compression [46] generalizes this to non-Gaussian data and parameter-dependent covariances through a two-step procedure: first, extract intermediate summaries from raw data (e.g., power spectra or principal components); second, linearly compress these summaries to the score function – the gradient of an approximate log-likelihood with respect to parameters. This two-step approach allows for non-linear feature extraction in the first stage while maintaining asymptotic optimality through score compression in the second. Information Maximizing Neural Networks (IMNN) [47] perform non-linear compression by training neural networks to directly maximize the determinant of the Fisher information matrix, learning optimal summaries from simulations and their derivatives without requiring intermediate hand-crafted features. The trade-off is between the modularity and interpretability of two-step approaches versus the flexibility of end-to-end learned neural compression.

4.3 End-to-End Neural Compression

Alternatively, rather than compressing data before inference, compression and inference networks can be trained simultaneously in a unified framework, leveraging the ability of modern deep neural networks to learn representations from high-dimensional data. Consider NPE as an example:

the complete architecture consists of two components: an embedding network s_Ψ that transforms raw simulations into learned representations, followed by a conditional density estimator q_Φ that maps these representations to posterior distributions. During training, gradients from the inference objective (e.g., the NPE loss in Equation 9) flow back through both networks, allowing the embedding network to automatically discover features useful for parameter inference without manual feature engineering. The embedding network architecture should be chosen based on data structure: e.g. convolutional neural networks (CNNs) exploit spatial structure in images, recurrent neural networks (RNNs) capture temporal dependencies in sequences. This approach extends naturally to NRE, where embedding networks learn representations optimized for classification. For NLE, end-to-end learning of an embedding network is not stable: the likelihood objective provides no incentive for the embedding to preserve parameter-sensitive information, and empirical studies show that joint training tends to collapse to uninformative summaries, so compression must instead be pretrained or fixed before likelihood estimation (see e.g. [48]).

While end-to-end training can theoretically learn sufficient statistics with adequate data and model capacity, practical limitations include high sample requirements for training deep networks, reduced interpretability compared to physics-motivated summaries, and vulnerability to distribution shift when observed data differs from the training simulations.

5 Marginal Inference

In many scientific applications, we are ultimately interested in inferring only a subset of model parameters while treating the rest as nuisances to be marginalized over. In each case, the parameters of scientific interest represent only a low-dimensional subspace of the full parameter space, but accurate inference requires marginalizing over nuisance spaces.

Formally, we partition the full parameter space $\Theta = \{\theta, \eta\} \in \mathbb{R}^{D_\Theta}$ into parameters of interest $\theta \in \mathbb{R}^{D_\theta}$ and nuisance parameters $\eta \in \mathbb{R}^{D_\Theta - D_\theta}$, where typically $D_\theta \ll D_\Theta$. The marginal posterior is obtained by integrating the joint posterior over all nuisance parameters:

$$p(\theta \mid \boldsymbol{x}) = \int \mathrm{d}\eta \, p(\theta, \eta \mid \boldsymbol{x}) \; . \tag{26}$$

In likelihood-based inference, this marginalization integral can be analytically intractable or computationally prohibitive, limiting the complexity of models that can be analyzed in practice. But one of the key advantages of SBI is that marginal inference can be performed naturally and efficiently, without requiring explicit evaluation of the high-dimensional integral.

Across all seen SBI methods, marginalization is achieved through a simple but powerful principle: nuisance parameters are *sampled during training* but *not passed* to the neural network [19, 30, 49]. In practice, training data is generated by simulating with both parameters of interest θ and nuisance parameters η, but the neural networks (classifiers for NRE, density estimators for NPE and NLE) are trained only on $(\boldsymbol{x}, \theta)$ pairs. By integrating over η in the training objective loss functions, the networks learn to correctly approximate marginal quantities: the marginal ratio $r(\theta; \boldsymbol{x}) \approx d_\Phi(\boldsymbol{x}, \theta)/(1 - d_\Phi(\boldsymbol{x}, \theta))$ for NRE, the marginal posterior $p(\theta \mid \boldsymbol{x}) \approx q_\Phi(\theta \mid \boldsymbol{x})$ for NPE, or the marginal likelihood $p(\boldsymbol{x} \mid \theta) \approx q_\Phi(\boldsymbol{x} \mid \theta)$ for NLE. Since the data $\boldsymbol{x}$ implicitly incorporates variance from nuisance parameters, the neural networks learn the correct marginal relationships without ever computing high-dimensional integrals.

This capability to directly target marginal posteriors is important for high-dimensional problems. Rather than first inferring a D_Θ-dimensional joint posterior and then numerically marginalizing,

SBI methods sidestep the curse of dimensionality by never representing the nuisance dimensions explicitly (see e.g. [50]). This allows to scale model complexity (adding more nuisance parameters for realism) without proportionally increasing inference cost, as long as the marginal data variance remains informative about the parameters of interest.

6 Sequential Inference

Active learning improves sample efficiency by using acquired knowledge to guide where the simulator is run next. The key idea is to iteratively simulate at parameter points expected to maximize information gain, updating our knowledge after each simulation to inform the next choice. In Bayesian inference, this translates to continuously updating the posterior and using it to steer the proposal distribution for simulator parameters [51].

Within SBI, *sequential* methods implement active learning by adaptively refining the proposal distribution for model parameters across multiple rounds of simulation. Sequential approaches have demonstrated greater simulation efficiency compared to non-sequential methods across benchmark tasks [16] and in applications ranging from cosmology [50] to gravitational wave astrophysics [52] and strong gravitational lensing [53].

The motivation for sequential SBI is straightforward: when targeting a specific observation $\boldsymbol{x}_0$, sampling parameters uniformly from the prior is inefficient. Training data $(\boldsymbol{x}, \Theta)$ generated from the prior exhibits high variance relative to $\boldsymbol{x}_0$, meaning that for a fixed simulation budget, relatively few samples provide information about the target posterior $p(\Theta \mid \boldsymbol{x}_0)$. Sequential algorithms address this by drawing parameters from an adaptive proposal distribution $p^{(R)}(\Theta)$ in each round R, progressively focusing simulations on parameter regions consistent with $\boldsymbol{x}_0$.

In SBI, there are multiple technical realizations of this idea of active learning, and various ways of choosing the adaptive proposal $p^{(R)}(\Theta)$. The main sequential approaches include using the posterior from the previous round as the proposal distribution, with various schemes to correct for the resulting distribution mismatch (as in sequential NPE methods, SNPE-A/B/C) [23, 54, 24], and restricting the prior support to high-posterior/likelihood-density regions discovered in earlier rounds (prior truncation) [55, 56, 57]. Recently, a dynamic SBI algorithm has been proposed, which implements the core ideas of sequential methods in a round-free, asynchronous, and parallelisable manner [58].

The general ways to define a proposal distribution are illustrated in Figure 6. The left panel shows box prior truncation, the middle panel shows correlated prior truncation, while the right panel shows a general proposal distribution proportional to the posterior. We now discuss them.

6.1 Prior truncation

Prior truncation implements active learning by restricting the prior support to regions of high posterior/likelihood density identified in previous rounds [55, 56, 57].

Broadly, prior truncation schemes iteratively discard in rounds R low likelihood/posterior/ratio regions, where the current approximate of the likelihood/posterior/ratio evaluated for the target observation is below a user defined threshold ϵ. Concretely, in the case of NPE, this means keeping the region of parameter space defined by

$$\Gamma_{\Theta}^{(R)} = \{\Theta \in \mathbb{R}^{D_\Theta} : p^{(R)}(\Theta \mid \boldsymbol{x}) > \epsilon\} \,. \tag{27}$$

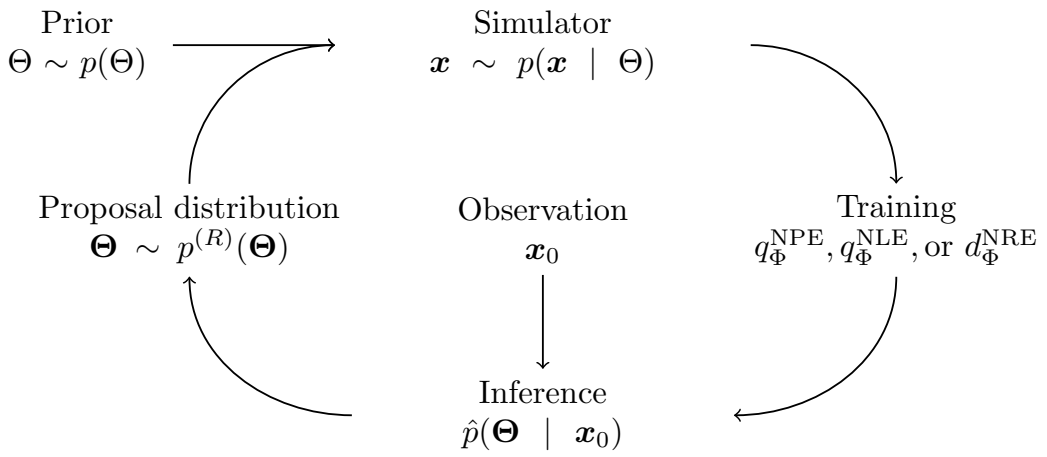

Figure 5: **Sequential SBI**. Parameters Θ are sampled from the initial prior $p(\Theta)$, and mapped into data $\boldsymbol{x}$ through a programmed simulator that effectively acts as an implicit likelihood. Inference is performed by training an SBI algorithm, whether NPE (q_Φ^{NPE}), NLE (q_Φ^{NLE}), or NRE (d_Φ^{NRE}) to learn the estimates $\hat{p}(\Theta; \boldsymbol{x})$ of the posterior of interest. The ratios are used to weight prior samples, enabling posterior sampling for any observation $\boldsymbol{x}$. Focusing on a target observation $\boldsymbol{x}_0$, we can perform sequential inference updating the proposal distribution $p^{(R)}(\Theta)$, by constraining the initial parameter space for the inferred parameters to regions that are most relevant for a specific target observation $\boldsymbol{x}_0$. The procedure is repeated until convergence.

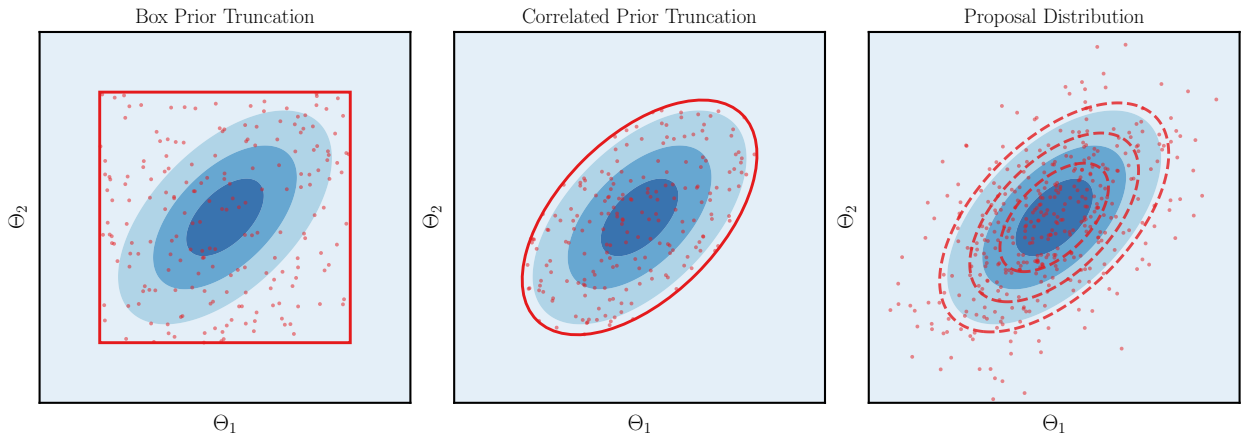

Figure 6: **Proposal distributions in sequential SBI.** Comparison of strategies for focusing simulations on regions relevant to observation $\boldsymbol{x}_0$ (blue: true posterior; red: proposal). *Left:* Box truncation samples uniformly within a hyper-rectangular region. *Center:* Correlated prior truncation samples uniformly within iso-likelihood/posterior contours. *Right:* Learned proposal distribution approximates the posterior from the previous round.

Importantly, since this sequential scheme does not modify the shape of the prior proposal distribution, but only restricts its support, the inference is still locally amortised in the constrained proposal distribution region. This enables empirical tests of the inference result (see Section 7).

A common realization is *box truncation*, where the prior is restricted to a hyper-rectangular region in parameter space [55]. This is achieved by thresholding one-dimensional marginal posteriors at a small fraction ϵ of their maximum density, effectively defining bounds for each parameter independently. The threshold ϵ is typically chosen as to truncate at approximately $\pm 5\sigma$ for Gaussian posteriors, as to preserve the bulk of the posterior mass while excluding negligible tails. The truncation is refined iteratively across multiple rounds using nested indicator functions[7]:

$$\tilde{p}^R(\boldsymbol{\Theta}) = \frac{1}{Z}\mathbb{I}(\Theta_1 \in \Gamma_{\Theta_1}^{(R-1)}) \times \cdots \times \mathbb{I}(\Theta_d \in \Gamma_{\Theta_d}^{(R-1)}) p(\boldsymbol{\Theta})\,, \tag{29}$$

Box truncation is simple to implement but can be inefficient in high-dimensional spaces with strong parameter correlations, as it may include excessive low-density regions. Alternative methods extend beyond boxes to more flexibly restrict the prior support [56, 57]. In these cases, the prior is truncated to the highest-probability region (HPR) containing $1-\epsilon$ of the approximate posterior/likelihood/ratio mass from the previous round. This is implemented via rejection-sampling or nested-sampling from the full prior and accepting only samples where the log-density exceeds the ϵ-quantile threshold,

$$\tilde{p}^R(\Theta) = \frac{1}{Z}\mathbb{I}(\Theta \in \Gamma_{\Theta}^{(R-1)}) p(\Theta)\,. \tag{30}$$

This implementation results in a tighter confinement to high-density regions than in box truncation, improving simulation efficiency in exploring complex, correlated posteriors.

6.2 Sequential NPE

In general, sequential NPE (SNPE) algorithms iteratively replace the prior $p(\Theta)$ with a suitably chosen proposal distribution $\tilde{p}(\Theta)$ (usually proportional to the current estimate of the posterior) that focuses simulation on parameter regions relevant to the observation of interest $\boldsymbol{x}_0$. However, when the proposal distribution differs from the prior, a problem arises: if the standard NPE loss from Equation (9) is applied without modification, the network learns to approximate a proposal-biased posterior:

$$q_\Phi^{\mathrm{NPE}}(\Theta \mid \boldsymbol{x}) \simeq \frac{1}{Z}\, p(\boldsymbol{x} \mid \Theta)\tilde{p}(\Theta)\,, \tag{31}$$

where Z is a normalisation constant. Since the goal of all SNPE algorithms is to recover the true posterior $p(\Theta \mid \boldsymbol{x}) \propto p(\boldsymbol{x} \mid \Theta)p(\Theta)$, each SNPE variant (SNPE-A/B/C) employs a different strategy to correct for the proposal bias, differing in what is targeted by q_Φ and which NPE loss variant is used.

6.2.1 SNPE-A

In the SNPE-A [23] version of sequential SBI, the analysis is carried out in multiple rounds R, with proposal distributions initially set to be the prior $\tilde{p}^{(R=1)}(\Theta) = p(\Theta)$ and then iterative approxima-

[7] Here, $\mathbb{I}_\Gamma(\Theta)$ is an indicator function, which is one for $\Theta \in \Gamma$ and zero otherwise,

$$\mathbb{I}_\Gamma(\Theta) \equiv \begin{cases} 1 & \text{for} \quad \Theta \in \Gamma \\ 0 & \text{otherwise} \end{cases} \tag{28}$$

tions to the posterior evaluated on the observation of interest $\boldsymbol{x}_0$,

$$\tilde{p}^{(R)}(\Theta) = q_\Phi^{(R-1)}(\Theta \mid \boldsymbol{x}_0) \,. \tag{32}$$

After training on the loss in Equation (9) for R rounds, the true posterior is related to the learned approximate posterior $q_\Phi^{(R)}(\Theta \mid \boldsymbol{x})$ via the following

$$p(\Theta \mid \boldsymbol{x}) \propto q_\Phi^{(R)}(\Theta \mid \boldsymbol{x}) \frac{p(\Theta)}{\tilde{p}^{(R)}(\Theta)} \,. \tag{33}$$

We see that this strategy typically learns a distribution $q_\phi^{(R)}$ that is narrower than the true posterior, since the proposal concentrates on high-likelihood regions. The bias can be corrected by importance reweighting: taking samples from $q_\Phi^{(R)}(\Theta \mid \boldsymbol{x}_0)$ and reweighting them according to the ratio $\frac{p(\Theta)}{\tilde{p}^{(R)}(\Theta)}$ to recover the true posterior as in Equation (33).

The traditional SNPE-A algorithm uses Gaussian mixture density networks, single Gaussians proposals, and Gaussian or uniform priors. This choice enables a closed-form test-time correction $q_\Phi(\Theta \mid \boldsymbol{x}_0)\, p(\Theta)/\tilde{p}(\Theta)$ to be analytically normalizable and directly samplable for every round, but it also limits flexibility: Gaussian proposals struggle with multi-modality or heavy tails and can induce mode collapse or poor coverage when the proposal has weak overlap with the true posterior.

6.2.2 SNPE-B

SNPE-B [54] takes a different approach by correcting for proposal bias directly in the training objective rather than at test time, by modifying the NPE loss via importance weighting. Specifically, each training sample $(\boldsymbol{x}, \Theta)$ drawn from the proposal $\tilde{p}(\Theta)$ is weighted by the ratio $p(\Theta)/\tilde{p}(\Theta)$ during training:

$$\mathcal{L}^{\text{SNPE-B}}[\Phi] = -\mathbb{E}_{\Theta \sim \tilde{p}(\Theta), \boldsymbol{x} \sim p(\boldsymbol{x}\mid\Theta)} \left[\frac{p(\Theta)}{\tilde{p}(\Theta)} \log q_\Phi^{\text{NPE}}(\Theta \mid \boldsymbol{x}) \right] \,. \tag{34}$$

This importance weighting corrects for the mismatch between the proposal and the prior during training. As a result, the learned distribution q_Φ^{NPE} directly approximates the true posterior, $q_\Phi^{\text{NPE}}(\Theta \mid \boldsymbol{x}) \simeq p(\Theta \mid \boldsymbol{x})$, requiring no reweighting during sampling. This eliminates the need for test-time corrections and removes the restriction to Gaussian proposals, allowing for more expressive proposal distributions such as those parameterized by normalizing flows. However, the importance weights can have high variance during training, leading to inaccurate inference for some tasks.

6.2.3 SNPE-C

SNPE-C, also known as Automatic Posterior Transformation (APT) [24], addresses the high variance issues that can arise from the importance weighting in SNPE-B while maintaining its flexibility. The main innovation is a reparameterization that allows training on an approximate proposal posterior rather than directly on the true posterior. Instead of using importance weights during training, SNPE-C trains the density estimator $q_\Phi(\Theta \mid \boldsymbol{x})$ to maximize the likelihood under a modified objective that accounts for the proposal distribution through a clever sampling scheme. The loss is computed over "proposal sets" Θ sampled from a hyper-proposal distribution $V(\Theta)$, which ensures that the trained network learns the correct posterior despite being trained on data generated from the proposal $\tilde{p}(\Theta)$.

A major advantage of SNPE-C over SNPE-A is the introduction of "atomic" proposals, which remove restrictions on the density estimator architecture. This allows the use of highly expressive models such as normalizing flows without requiring analytical tractability for the test-time correction. Additionally, SNPE-C can use data from all previous rounds by reweighting past simulations, improving sample efficiency compared to methods that discard earlier rounds. In practice, SNPE-C combines the advantages of both predecessors: like SNPE-B, it directly approximates the true posterior without post-hoc corrections, and like SNPE-A, it avoids the high-variance importance weights during training.

7 Diagnostics and tests

Assessing the quality of SBI posteriors is extremely important, since neural networks can fail silently without proper validation. Unlike traditional MCMC where convergence diagnostics provide some reassurance, neural SBI methods require specialized tests to verify that estimated posteriors are statistically calibrated and accurate. These diagnostics only need the simulator and evaluate statistical consistency across many simulated observations.

7.1 Posterior predictive checks

The simplest validation approach is a posterior predictive check: draw parameters from the estimated posterior, simulate data, and compare these samples to the observed data, i.e.

$$p(\boldsymbol{x} \mid \boldsymbol{x}_0) = \int p(\boldsymbol{x} \mid \Theta)\, p(\Theta \mid \boldsymbol{x}_0)\, \mathrm{d}\Theta \,. \tag{35}$$

If the simulated and observed data distributions match, the posterior captures the data-generating process. However, this check is necessary but not sufficient: an uninformative posterior (e.g., the prior) can pass predictive checks while providing no useful parameter constraints.

7.2 Inference robustness checks

Inference should be stable under small, structured changes to the data or the inference pipeline. Useful robustness checks include: masking parts of the observed data to see if inference relies too heavily on specific features; changing slightly the summary statistic or embedding network architecture to test sensitivity to learned representations; and altering training seeds, optimizers, or architectures to probe stability under different conditions.

Of course, these tests come with caveats. Variability in results could reflect training instability, a poorly tuned network, or model misspecification. These stress tests do not replace more principled diagnostics, but they are lightweight, easy to run, and useful for early warnings.

7.3 Coverage diagnostics

More rigorous validation comes from coverage diagnostics, which test whether posterior credible regions have correct coverage properties on average across many observations. These diagnostics require generating a calibration set of L parameter-simulation pairs $\{(\Theta_i, \boldsymbol{x}_i)\}_{i=1}^{L}$ drawn from the joint distribution $p(\Theta, \boldsymbol{x})$. For each pair, inference produces an estimated posterior $\hat{p}(\Theta \mid \boldsymbol{x}_i)$, which is then compared against the true parameter Θ_i that generated the simulation.

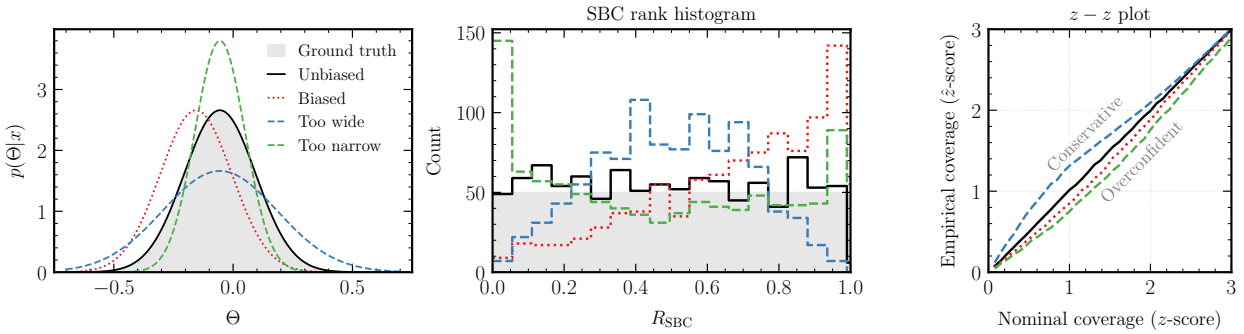

Figure 7: **SBI coverage diagnostics**. *Left*: Different posterior approximations for a one-dimensional parameter Θ, including the ground truth (gray shaded), an unbiased estimator (black), a biased estimator with shifted mean (red dotted), an overly conservative estimator with increased variance (blue dashed), and an overconfident estimator with reduced variance (green dashed). *Middle*: SBC rank histograms testing uniformity of rank statistics. A well-calibrated posterior (unbiased, black) produces a uniform distribution matching the ground truth expectation (gray shaded). Biased posteriors show skewed distributions, while miscalibrated uncertainties produce U-shaped (too narrow) or peaked (too wide) histograms. *Right*: Expected coverage test displayed as a z-z plot comparing nominal versus empirical coverage. Perfect calibration follows the diagonal (ground truth). Conservative posteriors (too wide) lie above the diagonal, while overconfident posteriors (too narrow) fall below.

All coverage methods use a ranking procedure. For each calibration sample i, draw M posterior samples $\{\Theta_i^{(m)}\}_{m=1}^M \sim \hat{p}(\Theta \mid \boldsymbol{x}_i)$ and compute a rank based on how the true parameter Θ_i compares to these samples. The rank is defined for a projection $F : \Theta \to F(\Theta) \in \mathbb{R}$ as:

$$R_i = \frac{1}{M} \sum_{m=1}^{M} \mathbb{I}[F(\Theta_i^{(m)}) < F(\Theta_i)] \in [0, 1], \tag{36}$$

where $\mathbb{I}[\cdot]$ is the indicator function. If the posterior is well-calibrated, these ranks should be uniformly distributed across the calibration set.

The most used coverage diagnostics are illustrated in Figure 7. The left panel shows different types of posterior approximations: unbiased, biased (shifted mean), and miscalibrated (incorrect uncertainty). The middle panel demonstrates simulation-based calibration, while the right panel shows expected coverage tests. We now discuss these diagnostics.

Simulation-Based Calibration Simulation-based calibration (SBC) [59] applies the ranking procedure to each parameter dimension separately, setting $F(\Theta) = \Theta_j$ for the j-th parameter. For each dimension, the histogram of ranks across all L calibration samples should be uniform on $[0, 1]$. Deviations indicate miscalibration: a U-shaped histogram suggests the posterior is too narrow (overconfident), while an inverted-U shape indicates it is too broad (underconfident). Skewed histograms reveal systematic bias toward high or low parameter values. SBC can identify which specific parameters have calibration issues but cannot detect problems in parameter correlations if all marginals are individually well-calibrated.

Expected Coverage Expected coverage tests [60] the joint posterior by using the log-probability as the ranking function: $F(\Theta) = \log \hat{p}(\Theta \mid \boldsymbol{x})$. This defines highest posterior density (HPD) credible

regions: the $(1-\alpha)$ HPD region $\Omega_{\hat{p}}(1-\alpha)$ contains all parameters with $\log \hat{p}(\Theta \mid \boldsymbol{x}) \geq 1-\alpha$, where c_α is chosen such that $\int_{\Omega_{\hat{p}}(1-\alpha)} \hat{p}(\Theta \mid \boldsymbol{x}) \mathrm{d}\Theta = 1-\alpha$. The empirical coverage is:

$$1 - \hat{\alpha} = \mathbb{E}_{p(\boldsymbol{x},\Theta)} \left[\mathbb{I} \left[\Theta \in \Omega_{\hat{p}(\Theta|\boldsymbol{x})}(1-\alpha) \right] \right], \tag{37}$$

estimated across the calibration set. A well-calibrated posterior satisfies $1 - \hat{\alpha} = 1 - \alpha$ for all α. This is usually visualized on probability-probability (P-P) plots, where nominal coverage $1 - \alpha$ is plotted against empirical coverage $1 - \hat{\alpha}$. Alternatively, since we are mostly interested in small values of α, corresponding to posterior regions with high mass, it is convenient to reparameterize α in terms of a new variable z, defined as $1 - \frac{1}{2}\alpha$ quantile of the standard normal distribution (see e.g. [50]). As a results, $z = 1, 2, 3$ have $1 - \alpha = 0.6827, 0.9545, 0.9997$, and correspond to the common 1σ, 2σ, 3σ regions. This visualization is usually called z-score–z-score (Z-Z) plot. In both types of visualization, perfect calibration yields a diagonal line; conservative posteriors (too wide) appear above the diagonal, while overconfident posteriors (too narrow) fall below. Because expected coverage tests the joint distribution, it can detect miscalibration in parameter correlations.

Agreement between the nominal and empirically-measured expected coverage is a necessary (but not sufficient) condition for the density/ratio estimator to be a correct estimate of the posterior. Indeed, the expected coverage test checks for the consistency of the posterior, not for its precision. For example, a not focused posterior that resembles the prior leads to a diagonal plot with optimal coverage. Moreover, it tests for the coverage *averaged* over parameter values $\Theta \sim p(\Theta)$. As a result, some parameter regions might be over- and other parameter regions under-covered.

Tests of Accuracy with Random Points (TARP) TARP [61] constructs credible regions as balls centered at randomly chosen reference points Θ_{ref} throughout parameter space, with radii extending to the true parameter. Specifically, for each calibration sample, TARP samples a reference point $\Theta_{\text{ref}} \sim \hat{p}(\Theta \mid \boldsymbol{x}_i)$ and checks whether a random posterior sample $\Theta' \sim \hat{p}(\Theta \mid \boldsymbol{x}_i)$ is closer to Θ_{ref} than the true parameter Θ_i. By varying the reference point distribution and checking coverage across different credibility levels, TARP can detect systematic biases that HPD-based methods might miss and works with generative models where explicit density evaluations are intractable.

Computational considerations about coverage tests Coverage diagnostics require performing inference on hundreds to thousands of calibration samples, making computational cost a critical factor. NPE enables rapid validation by directly producing posterior samples via forward passes, while NLE and NRE require MCMC sampling for each calibration point, often making comprehensive testing slow. For sequential methods like SNPE-A/B/C that use proposal distributions, these diagnostics become infeasible as the learned posteriors are valid only for specific observations; however, methods with prior truncation maintain locally amortized inference within the truncated support, enabling validation in each sequential round.

8 Good practices

All the elements we have seen so far form the backbone of an SBI analysis workflow (Figure 8). We will now discuss some good practices to keep in mind when approaching an SBI analysis.

8.1 When to use SBI

If the likelihood function is analytically tractable and computationally feasible to evaluate, traditional sampling-based inference methods remain the gold standard. These methods are well-

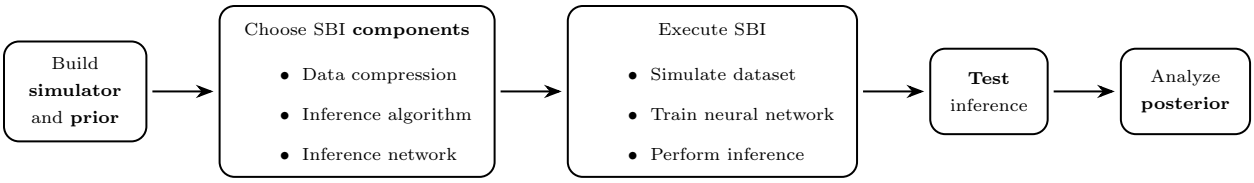

Figure 8: **SBI workflow overview.** (1) Define the inference problem, which includes setting up the simulator and the prior. (2) Choose appropriate components for performing SBI: a representation of the data (Section 4), an inference algorithm (e.g., NPE, NLE, NRE) and associated inference network (Section 3). (3) Run SBI: generate the simulated dataset, train the inference network, and, after training, evaluate the inference network to infer the posterior given an observation. (4) Validate the inference results with diagnostic tools (Section 7). (5) If these checks pass, analyze the posterior and simulator to gain scientific insight. **Credits:** Adapted from [17, Figure 3].

established, extensively validated, and avoid the complexities of training and validating neural networks. SBI should not be viewed as a replacement for likelihood-based inference when likelihoods are available, but it is a tool for settings where such methods become impractical or impossible.

Expensive or intractable likelihoods SBI becomes advantageous when likelihoods cannot be written down analytically or are prohibitively expensive to evaluate. This commonly arises when complex simulators involve implicit marginalization over many latent variables, or when the forward model is a black-box code.

Amortization Even when likelihoods are available, SBI offers practical advantages through amortization. Traditional methods require rerunning expensive MCMC chains from scratch for each new observation, a big bottleneck for experiments requiring real-time analysis. Amortized SBI trains once on simulations, then performs fast inference on new observations.

Avoid lossy compressions With the possibility of learning compressions via embedding networks (Section 4), SBI can extract information directly from high-dimensional data without relying on hand-crafted summary statistics that may loose useful information. This flexibility to work with complex data structures while scaling to high-dimensional parameter spaces makes SBI particularly powerful for modern scientific applications where both data and models are increasingly complex.

Curse of dimensionality SBI also helps overcome the curse of dimensionality in high-parameter models, by directly targeting marginal posteriors by training on parameters of interest while sampling over nuisances during simulation (Section 5). This avoids the exponential cost scaling that plagues MCMC in high dimensions when only low-dimensional marginals are of interest.

Use with caution While SBI offers great opportunities, it is not a plug-and-play solution. Successful application requires careful consideration of interconnected design choices: data compression, network architecture, training procedures, sequential strategies, and validation. Each component introduces potential failure modes: from poorly chosen summary statistics that discard information, to undertrained networks that yield miscalibrated posteriors, to inappropriate sequential strategies that introduce bias. Significant effort must be invested in understanding the method's assumptions and developing intuition for when the approach is working correctly. SBI should be viewed as a sophisticated tool requiring expertise, not as a black-box alternative to traditional inference.

8.2 Choosing SBI algorithm

The three neural SBI algorithms described in Section 3 have their distinct benefits and drawbacks. The choice between NPE, NLE, and NRE depends on the problem's characteristics and inference requirements. Here, we summarize the main trade-offs; for more details see Refs. [16, 17].

Inference speed and amortization All three methods are amortized: once trained, they can perform inference on new observations without additional simulations. However, inference speed differs significantly. NPE directly samples from the posterior in milliseconds via a single forward pass, making it ideal for real-time analysis or processing thousands of observations. In contrast, NLE and NRE require MCMC or variational inference to sample from the posterior, taking seconds to minutes depending on parameter dimensionality and introducing potential sampling errors.

Training complexity NRE trains a binary classifier, typically requiring fewer computational operations per training step than the conditional generative models used by NPE and NLE (e.g. normalizing flows). However, training cost rarely dominates the overall computational budget.

High-dimensional data The architectural differences between methods have implications for handling high-dimensional data. NPE and NRE take data $\boldsymbol{x}$ as input, naturally accommodating high-dimensional observations through embedding networks. These embeddings can be trained end-to-end with the inference network, automatically learning informative features. NLE takes parameters Θ as input to predict data, fundamentally limiting its ability to use embedding networks. For high-dimensional raw data, prefer NPE or NRE; consider NLE only for low-dimensional data or validated summary statistics.

Multiple observations A key advantage of NLE and NRE is that they can handle multiple independent and identically distributed (i.i.d.) observations $\{\boldsymbol{x}_i\}_{i=1}^{M}$, such as independent trials within a single experiment. In this case, the total likelihood for a set of observations can by exploiting the factorization $p(\{\boldsymbol{x}_i\} \mid \Theta) = \prod_i p(\boldsymbol{x}_i \mid \Theta)$. NLE and NRE can train on single observations yet perform inference on any number of observations. NPE must train on M observations to infer from M observations, requiring M simulations per parameter set and making it more simulation-intensive in this scenario.

Marginalization and sequential inference All three methods can directly target marginal posteriors by excluding nuisance parameters from the network input while including them in simulations (Section 5). All can also employ sequential inference to improve sample efficiency (Section 6). However, combining marginalization with sequential NPE requires care, as the proposal distribution affects the marginal posterior in ways that cannot be easily corrected post-training [62].

References

[1] W. K. Hastings. Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika*, 57:97–109, 1970. doi:10.1093/biomet/57.1.97.

[2] J. Skilling. Nested Sampling. *AIP Conf. Proc.*, 735(1):395, 2004. doi:10.1063/1.1835238.

[3] R. M. Neal. MCMC using hamiltonian dynamics. 2012. arXiv:1206.1901.

[4] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 2013. arXiv:1206.7051.

[5] K. Cranmer, J. Brehmer, and G. Louppe. The frontier of simulation-based inference. *Proc. Nat. Acad. Sci.*, 117(48):30055–30062, 2020. arXiv:1911.01429, doi:10.1073/pnas.1912789117.

[6] D. B. Rubin. Bayesianly justifiable and relevant frequency calculations for the applied statistician. *The Annals of Statistics*, pages 1151–1172, 1984. doi:10.1214/aos/1176346785.

[7] J.M. Marin, P. Pudlo, C. P. Robert, and R. J. Ryder. Approximate bayesian computational methods. *Statistics and computing*, 22(6):1167–1180, 2012. arXiv:1101.0955.

[8] S. A. Sisson, Y. Fan, and M. A. Beaumont. Overview of approximate bayesian computation. 02 2018. arXiv:1802.09720.

[9] C. Grazian and Y. Fan. A review of approximate bayesian computation methods via density estimation: inference for simulator-models. 09 2019. arXiv:1909.02736.

[10] E. E. O. Ishida, S. D. P. Vitenti, M. Penna-Lima, J. Cisewski, R. S. de Souza, A. M. M. Trindade, E. Cameron, and V. C. Busti. *COSMOABC*: Likelihood-free inference via Population Monte Carlo Approximate Bayesian Computation. *Astron. Comput.*, 13:1–11, 2015. arXiv:1504.06129, doi:10.1016/j.ascom.2015.09.001.

[11] J. Akeret, A. Refregier, A. Amara, S. Seehars, and C. Hasner. Approximate Bayesian Computation for Forward Modeling in Cosmology. *JCAP*, 08:043, 2015. arXiv:1504.07245, doi:10.1088/1475-7516/2015/08/043.

[12] E. Jennings and M. Madigan. astroabc: an approximate bayesian computation sequential monte carlo sampler for cosmological parameter estimation. *Astronomy and computing*, 19:16–22, 2017. arXiv:1608.07606, doi:10.1016/j.ascom.2017.01.001.

[13] F. Leclercq. Bayesian optimization for likelihood-free cosmological inference. *Phys. Rev. D*, 98(6):063511, 2018. arXiv:1805.07152, doi:10.1103/PhysRevD.98.063511.

[14] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521:436–44, 05 2015. doi:10.1038/nature14539.

[15] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. Automatic differentiation in machine learning: a survey. *Journal of machine learning research*, 18(153):1–43, 2018. arXiv:1502.05767.

[16] Jan-Matthis Lueckmann, Jan Boelts, David S. Greenberg, Pedro J. Gonçalves, and Jakob H. Macke. Benchmarking simulation-based inference. 01 2021. arXiv:2101.04653.

[17] M Deistler, J. Boelts, P. Steinbach, G. Moss, T. Moreau, M. Gloeckler, P. L. C. Rodrigues, J. Linhart, J. K. Lappalainen, B. K. Miller, P. J. Gonçalves, J.M. Lueckmann, C. Schröder, and J. H. Macke. Simulation-based inference: A practical guide, 2025. URL: https://arxiv.org/abs/2508.12939, arXiv:2508.12939.

[18] M Ho et al. LtU-ILI: An All-in-One Framework for Implicit Inference in Astrophysics and Cosmology. *Open J. Astrophys.*, 7:001c.120559, 2024. arXiv:2402.05137, doi:10.33232/001c.120559.

[19] J. Alsing, T. Charnock, S. Feeney, and B. Wandelt. Fast likelihood-free cosmology with neural density estimators and active learning. *Mon. Not. Roy. Astron. Soc.*, 488(3):4440–4458, 2019. arXiv:1903.00007, doi:10.1093/mnras/stz1960.

[20] A. Tejero-Cantero, J. Boelts, M. Deistler, J.M. Lueckmann, C. Durkan, P. J. Gonçalves, D. S. Greenberg, and J. H. Macke. sbi: A toolkit for simulation-based inference. *Journal of Open Source Software*, 5(52):2505, 2020. doi:10.21105/joss.02505.

[21] Benjamin Kurt Miller, Alex Cole, Christoph Weniger, Francesco Nattino, Ou Ku, and Meiert W. Grootes. swyft: Truncated marginal neural ratio estimation in python. *Journal of Open Source Software*, 7(75):4205, 2022. doi:10.21105/joss.04205.

[22] F. Rozet and A. Delaunoy. LAMPE: Likelihood-free AMortized Posterior Estimation, 2022. URL: https://github.com/probabilists/lampe.

[23] G. Papamakarios and I. Murray. Fast ϵ-free Inference of Simulation Models with Bayesian Conditional Density Estimation. 5 2016. arXiv:1605.06376.

[24] David S. Greenberg, Marcel Nonnenmacher, and Jakob H. Macke. Automatic posterior transformation for likelihood-free inference. 05 2019. arXiv:1905.07488.

[25] George Papamakarios, David C. Sterratt, and Iain Murray. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. 05 2018. arXiv:1805.07226.

[26] Conor Durkan, George Papamakarios, and Iain Murray. Sequential neural methods for likelihood-free inference. 11 2018. arXiv:1811.08723.

[27] Kyle Cranmer, Juan Pavez, and Gilles Louppe. Approximating likelihood ratios with calibrated discriminative classifiers. 2015. arXiv:1506.02169.

[28] Michael U Gutmann, Ritabrata Dutta, Samuel Kaski, and Jukka Corander. Likelihood-free inference via classification. *Statistics and Computing*, 28:411–425, 2018. arXiv:1407.4981.

[29] Joeri Hermans, Volodimir Begy, and Gilles Louppe. Likelihood-free MCMC with Amortized Approximate Ratio Estimators. 3 2019. arXiv:1903.04057.

[30] Benjamin Kurt Miller, Alex Cole, Gilles Louppe, and Christoph Weniger. Simulation-efficient marginal posterior estimation with swyft: stop wasting your precious time. In *36th Conference on Neural Information Processing Systems*, 11 2020. arXiv:2011.13951.

[31] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951. doi:10.1214/aoms/1177729694.

[32] Christopher M. Bishop. Mixture density networks. 1994. URL: https://api.semanticscholar.org/CorpusID:118227751.

[33] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. 2016. URL: https://arxiv.org/abs/1505.05770, arXiv:1505.05770.

[34] Ivan Kobyzev, Simon JD Prince, and Marcus A Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):3964–3979, 2020. arXiv:1908.09257.

[35] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021. arXiv:1912.02762.

[36] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked Autoregressive Flow for Density Estimation. In *31st Conference on Neural Information Processing Systems*, 5 2017. arXiv:1705.07057.

[37] Diederik P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improving variational inference with inverse autoregressive flow. 2017. URL: https://arxiv.org/abs/1606.04934, arXiv:1606.04934.

[38] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. 2019. URL: https://arxiv.org/abs/1906.04032, arXiv:1906.04032.

[39] Cheng Zhang, Judith Bütepage, Hedvig Kjellström, and Stephan Mandt. Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):2008–2026, 2018. arXiv:1711.05597.

[40] Kevin P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022.

[41] Luc Devroye, László Györfi, and Gábor Lugosi. *A Probabilistic Theory of Pattern Recognition*, volume 31 of *Stochastic Modelling and Applied Probability*. Springer, 1996. doi:10.1007/978-1-4612-0711-5.

[42] Johann Brehmer, Siddharth Mishra-Sharma, Joeri Hermans, Gilles Louppe, and Kyle Cranmer. Mining for Dark Matter Substructure: Inferring subhalo population properties from strong lenses with machine learning. *Astrophys. J.*, 886(1):49, 2019. arXiv:1909.02005, doi:10.3847/1538-4357/ab4c41.

[43] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. 12 2015. arXiv:1512.03385, doi:10.1109/CVPR.2016.90.

[44] Anqi Mao, Mehryar Mohri, and Yutao Zhong. Cross-entropy loss functions: Theoretical analysis and applications. In *International conference on Machine learning*, pages 23803–23828. PMLR, 2023. arXiv:2304.07288.

[45] A. F. Heavens, R. Jimenez, and O. Lahav. Massive lossless data compression and multiple parameter estimation from galaxy spectra. *Monthly Notices of the Royal Astronomical Society*, 317(4):965–972, October 2000. URL: http://dx.doi.org/10.1046/j.1365-8711.2000.03692.x, doi:10.1046/j.1365-8711.2000.03692.x.

[46] J. Alsing, B. Wandelt, and S. Feeney. Massive optimal data compression and density estimation for scalable, likelihood-free inference in cosmology. *Mon. Not. Roy. Astron. Soc.*, 477(3):2874–2885, 2018. arXiv:1801.01497, doi:10.1093/mnras/sty819.

[47] Tom Charnock, Guilhem Lavaux, and Benjamin D. Wandelt. Automatic physical inference with information maximizing neural networks. *Physical Review D*, 97(8), April 2018. URL: http://dx.doi.org/10.1103/PhysRevD.97.083004, doi:10.1103/physrevd.97.083004.

[48] Niall Jeffrey, Justin Alsing, and Franç ois Lanusse. Likelihood-free inference with neural compression of DES SV weak lensing map statistics. *Mon. Not. Roy. Astron. Soc.*, 501(1):954–969, nov 2020. arXiv:2009.08459, doi:10.1093/mnras/staa3594.

[49] Niall Jeffrey and Benjamin D. Wandelt. Solving high-dimensional parameter inference: marginal posterior densities & Moment Networks. In *34th Conference on Neural Information Processing Systems*, 11 2020. arXiv:2011.05991.

[50] A. Cole, B. K. Miller, S. J. Witte, M. X. Cai, M. W. Grootes, F. Nattino, and C. Weniger. Fast and credible likelihood-free cosmology with truncated marginal neural ratio estimation. *JCAP*, 09:004, 2022. arXiv:2111.08030, doi:10.1088/1475-7516/2022/09/004.

[51] Edward Meeds and Max Welling. Optimization monte carlo: Efficient and embarrassingly parallel likelihood-free inference. 2015. URL: https://arxiv.org/abs/1506.03693, arXiv:1506.03693.

[52] Uddipta Bhardwaj, James Alvey, Benjamin Kurt Miller, Samaya Nissanke, and Christoph Weniger. Sequential simulation-based inference for gravitational wave signals. *Phys. Rev. D*, 108(4):042004, 2023. arXiv:2304.02035, doi:10.1103/PhysRevD.108.042004.

[53] Sebastian Wagner-Carena, Jaehoon Lee, Jeffrey Pennington, Jelle Aalbers, Simon Birrer, and Risa H. Wechsler. A strong gravitational lens is worth a thousand dark matter halos: Inference on small-scale structure using sequential methods. 2024. arXiv:2404.14487.

[54] J.M. Lueckmann, P. J. Goncalves, G. Bassetto, K. Öcal, M. Nonnenmacher, and J. H. Macke. Flexible statistical inference for mechanistic models of neural dynamics. In *31st Conference on Neural Information Processing Systems*, 11 2017. arXiv:1711.01861.

[55] B. K. Miller, A. Cole, P. Forré, G. Louppe, and C. Weniger. swyft: Truncated Marginal Neural Ratio Estimation in Python. *J. Open Source Softw.*, 7(75):4205, 2022. doi:10.21105/joss.04205.

[56] Michael Deistler, Pedro J Goncalves, and Jakob H Macke. Truncated proposals for scalable and hassle-free simulation-based inference. In *36th Conference on Neural Information Processing Systems*, 10 2022. arXiv:2210.04815.

[57] Noemi Anau Montel, James Alvey, and Christoph Weniger. Scalable inference with autoregressive neural ratio estimation. *Mon. Not. Roy. Astron. Soc.*, 530(4):4107–4124, 2024. arXiv:2308.08597, doi:10.1093/mnras/stae1130.

[58] Huifang Lyu, James Alvey, Noemi Anau Montel, Mauro Pieroni, and Christoph Weniger. Dynamic sbi: Round-free sequential simulation-based inference with adaptive datasets. 2025. URL: https://arxiv.org/abs/2510.13997, arXiv:2510.13997.

[59] Sean Talts, Michael Betancourt, Daniel Simpson, Aki Vehtari, and Andrew Gelman. Validating bayesian inference algorithms with simulation-based calibration, 2020. URL: https://arxiv.org/abs/1804.06788, arXiv:1804.06788.

[60] J. Hermans, A. Delaunoy, F. Rozet, A. Wehenkel, V. Begy, and G. Louppe. A Trust Crisis In Simulation-Based Inference? Your Posterior Approximations Can Be Unfaithful. *Transactions on Machine Learning Research*, 10 2022. arXiv:2110.06581, doi:https://openreview.net/forum?id=LHAbHkt6Aq.

[61] Pablo Lemos, Adam Coogan, Yashar Hezaveh, and Laurence Perreault-Levasseur. Sampling-based accuracy testing of posterior estimators for general inference. In *International Conference on Machine Learning*, pages 19256–19273. PMLR, 2023. arXiv:2302.03026.

[62] J. Alsing and B. Wandelt. Nuisance hardened data compression for fast likelihood-free inference. *Mon. Not. Roy. Astron. Soc.*, 488(4):5093–5103, 2019. arXiv:1903.01473, doi:10.1093/mnras/stz1900.