

LTU-ILI: AN ALL-IN-ONE FRAMEWORK FOR IMPLICIT INFERENCE IN ASTROPHYSICS AND COSMOLOGY

MATTHEW HO^{1*}, DEAGLAN J. BARTLETT¹, NICOLAS CHARTIER², CAROLINA CUESTA-LAZARO^{3,4,5}, SIMON DING¹, AXEL LAPEL^{1,14}, PABLO LEMOS^{6,7,8,9}, CHRISTOPHER C. LOVELL¹⁰, T. LUCAS MAKINEN¹¹, CHIRAG MODI^{9,12}, VIRAJ PANDYA^{13,15}, SHIVAM PANDEY¹³, LUCIA A. PEREZ⁹, BENJAMIN WANDELT^{1,9}, AND GREG L. BRYAN¹³

¹ CNRS & Sorbonne Université, Institut d'Astrophysique de Paris (IAP), UMR 7095, 98 bis bd Arago, F-75014 Paris, France

² Center for the Gravitational-Wave Universe, Astronomy Program Department of Physics and Astronomy, Seoul National University, Seoul 08826, Korea

³ Center for Astrophysics — Harvard & Smithsonian, 60 Garden St, Cambridge, MA 02138, USA

⁴ The NSF AI Institute for Artificial Intelligence and Fundamental Interactions

⁵ Department of Physics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA

⁶ Department of Physics, Université de Montréal, Montréal, Canada

⁷ Mila - Quebec Artificial Intelligence Institute, Montréal, Canada

⁸ Ciela - Montreal Institute for Astrophysical Data Analysis and Machine Learning, Montréal, Canada

⁹ Center for Computational Astrophysics, Flatiron Institute, 162 5th Avenue, New York, NY 10010, USA

¹⁰ Institute of Cosmology and Gravitation, University of Portsmouth, Burnaby Road, Portsmouth, PO1 3FX, United Kingdom

¹¹ Imperial Centre for Inference and Cosmology (ICIC) & Astrophysics Group, Imperial College London, Blackett Laboratory, Prince Consort Road, London SW7 2AZ, United Kingdom

¹² Center for Computational Mathematics, Flatiron Institute, 162 5th Avenue, New York, NY 10010, USA

¹³ Columbia Astrophysics Laboratory, Columbia University, 550 West 120th Street, New York, NY 10027, USA and

¹⁴ Sorbonne Université, Université Paris Diderot, Sorbonne Paris Cité, CNRS, Laboratoire de Physique Nucléaire et de Hautes Energies (LPNHE), 4 place Jussieu, F-75252, Paris Cedex 5, France

Version July 3, 2024

ABSTRACT

This paper presents the Learning the Universe Implicit Likelihood Inference (LTU-ILI) pipeline, a codebase for rapid, user-friendly, and cutting-edge machine learning (ML) inference in astrophysics and cosmology. The pipeline includes software for implementing various neural architectures, training schemata, priors, and density estimators in a manner easily adaptable to any research workflow. It includes comprehensive validation metrics to assess posterior estimate coverage, enhancing the reliability of inferred results. Additionally, the pipeline is easily parallelizable and is designed for efficient exploration of modeling hyperparameters. To demonstrate its capabilities, we present real applications across a range of astrophysics and cosmology problems, such as: estimating galaxy cluster masses from X-ray photometry; inferring cosmology from matter power spectra and halo point clouds; characterizing progenitors in gravitational wave signals; capturing physical dust parameters from galaxy colors and luminosities; and establishing properties of semi-analytic models of galaxy formation. We also include exhaustive benchmarking and comparisons of all implemented methods as well as discussions about the challenges and pitfalls of ML inference in astronomical sciences. All code and examples are made publicly available at <https://github.com/maho3/ltu-ili>.

Subject headings: cosmology, astrophysics, statistical methods, machine learning

1. INTRODUCTION

Statistical inference of unknown quantities is a fundamental problem in science. In the astronomical sciences, we largely rely on Bayesian inference to test new physical models (Feigelson and Babu 2012; Dodelson and Schmidt 2020; Eadie *et al.* 2023), wherein we assume some prior hypothesis of an observed system and calculate constraints on model parameters which would be consistent with our observations. For nearly a century, the practice of building linear or perturbative physical models from first-principles allowed us to make substantial progress towards understanding the universe. Yet, as highlighted in the 2020 Decadal Survey (National Academies of Sciences, Engineering, and Medicine 2023),

exploring the complex, nonlinear regime of physical phenomena through data-driven inference promises significant gains in constraining power. The large data volume of next-generation surveys, the improvements in high-resolution simulations, and the rapid rise of machine learning (ML) techniques have driven an explosion of inquiry into how one can automatically and rapidly learn complex physical phenomena (Carleo *et al.* 2019). Despite the growing enthusiasm, a major hurdle to the widespread adoption of data science methods in astronomy is in their limited accessibility to the general community, which when compounded by the rapid pace of developments in the field, makes it challenging for individuals to stay well-informed on best practices (Huppenkothen *et al.* 2023). There is currently no widespread consensus within the field regarding the approach to building a framework for implicit inference problems that is both

*matthew.ho@iap.fr

¹⁵ Hubble Fellow

reliable, robust, and accessible.

Implicit Likelihood Inference (ILI, Cranmer *et al.* 2020; Marin *et al.* 2012), also known as simulation-based inference (SBI) and likelihood-free inference (LFI), is an approach for learning the statistical relationship between parameters and data. ILI is a mathematically rigorous way of framing ML under the umbrella of Bayesian statistics, a language common to astrophysicists and cosmologists since the turn of the century (Christensen *et al.* 2001). Given a parameterized model and observed data, ILI estimates posterior distributions over model parameters, while accounting for predictive uncertainties. It is best explained in contrast to traditional explicit likelihood analyses, wherein one might write down an analytic likelihood to represent the probability of observations given a model and its parameters, and then use sampling methods such as Markov Chain Monte Carlo (MCMC; Brooks *et al.* 2011) to sample from the posterior. In ILI, one seeks to automatically learn the likelihood, i.e. to model the relationship between model parameters and observations through training on simulations. As such, ILI can be applied to infer parameters for any phenomenon that can be simulated, without the need for assumptions about the analytical form of the likelihood. Furthermore, ILI can accommodate complex data cuts (provided that consistent filters are applied to both the data and the forward model) something that can be hard to model in likelihood-based approaches.

Given a prescribed training set of model parameters and observations or a fast physical simulator, ILI can produce robust, tightly-constraining inference without analytic likelihoods. In astronomy and cosmology, ILI has consistently been shown to accelerate and improve upon traditional analyses such as cosmological galaxy lensing and clustering (*e.g.*, Jeffrey *et al.* 2021; Makinen *et al.* 2021, 2022a; de Santi *et al.* 2023a; Hahn *et al.* 2023a), gravitational waves (*e.g.*, Dax *et al.* 2021a; Cheung *et al.* 2022), galaxy cluster mass estimation (*e.g.*, Ho *et al.* 2022; de Andres *et al.* 2022), galaxy morphology (*e.g.*, Walmsley *et al.* 2020; Ghosh *et al.* 2022), stellar streams (*e.g.*, Hermans *et al.* 2021; Alvey *et al.* 2023), and exoplanets (*e.g.*, Rogers *et al.* 2023; Aubin *et al.* 2023).

Despite its popularity, robust applications of ILI have their challenges. First, fully Bayesian models include capturing epistemic (modeling) uncertainty, which requires marginalizing over model parameter uncertainty (*i.e.*, network weights and biases) and is intractable for large architectures. However, approximations to this method using dropout marginalization (Gal and Ghahramani 2016), model ensembling (Lakshminarayanan *et al.* 2017) or stochastic weight averaging (Maddox *et al.* 2019; Lemos *et al.* 2023) prove to be empirically sufficient surrogates. Second, choices of modeling hyperparameters (*e.g.*, preprocessing, architecture, learning procedures, etc.) can greatly affect the quality of a trained ILI model. This is generally addressed through model ensembling or hyperparameter searches (*e.g.*, Jin *et al.* 2019; White *et al.* 2021). Lastly, a pervasive problem in ILI is model misspecification, wherein the simulator or training catalog used to learn likelihoods is not representative of reality. Methods such as SBI++ (Wang *et al.* 2023) deal with missing data and outliers and present applications to galactic photometric redshift inference (Modi *et al.* 2023). For a thorough discussion of caveats, see Section

6. We argue that addressing these common problems is paramount to responsible application of ILI, and testing them exhaustively necessitates a unified, accessible framework.

We introduce the first version of the Learning the Universe Implicit Likelihood Inference code (LTU-ILI), a pipeline for training ML models for regressive parameter estimation. Given a labeled training set of observed data and parameters, LTU-ILI trains neural networks to emulate posterior probability distributions. The code also provides scientists with a diverse toolkit for testing and validation. It was built under the following design principles:

- The code includes state-of-the-art neural architectures, validation tools, and samplers for enabling ILI.
- The pipeline is modular, easily customizable, and parallelizable for rapid testing of design choices and hyperparameters.
- The interface is accessible to non-specialists while being practical for high-level production.
- The methodologies automatically implement standard best practices for machine learning (See Huppenkothen *et al.* 2023, for a review).

These attributes are paramount to make ILI a relevant tool for a wide range of inference problems, whether in the context of exploratory analysis or exhaustive empirical testing.

The novel contributions of this work are as follows:

- We unify several leading implicit inference codes—`sbi` (Tejero-Cantero *et al.* 2020), `pydelfi` (Alsing *et al.* 2019), and `lampe` (Rozet *et al.* 2021)—under a single common interface, for the first time allowing rigorous apples-to-apples comparisons of their performance.
- We supply extensions to these codes, enabling them to utilize new types of embedding networks (*e.g.*, for image- and graph-like datasets) and providing state-of-the-art validation metrics for enabling robust, maximally-informative inference.
- We provide an accessible, comprehensive, dual Jupyter and command-line interface, for rapid research and development.
- We demonstrate the abilities of ILI in an extensive selection of inference problems in astrophysics and cosmology, and we provide these as public benchmarks for future works.

At the time of publishing, LTU-ILI offers the broadest and most complete toolbox of features among ILI software. Over `sbi`, we include the capacity for custom data-loading procedures, additional neural density estimators (NDEs), and adaptability to exotic embedding architectures (including sequential and graph encoders). Over `pydelfi`, we allow the capacity for neural posterior and ratio estimation. Over `lampe`, we provide a configuration-based interface to training procedures, the

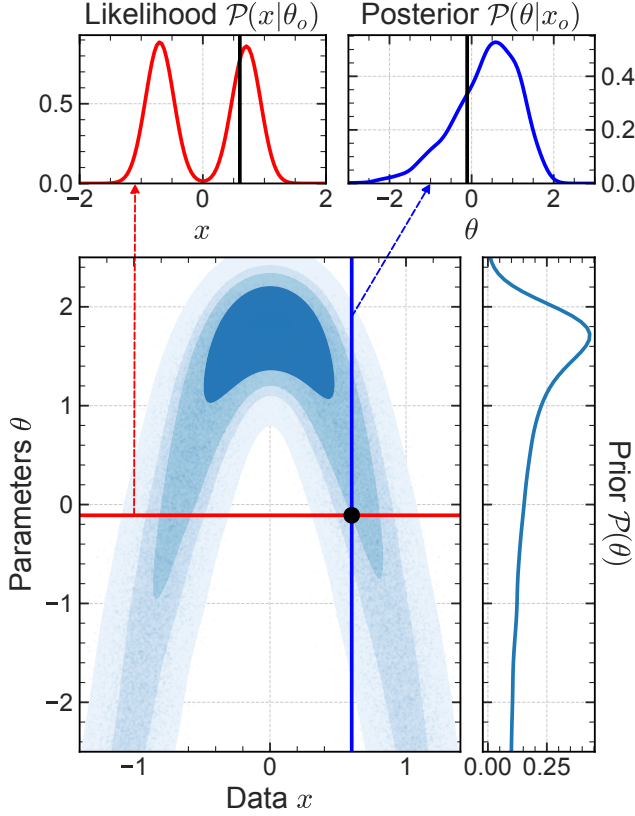


FIG. 1.— Demonstration of the differences between a likelihood (top-left), posterior (top-right), prior (bottom-right), and joint distribution (bottom-left) for an arbitrary one-dimensional inference problem. Here, the univariate data x and parameters θ are associated by a quadratic relationship with Gaussian scatter. Given the observed data point x_o (in black), our goal is to recover posterior constraints on θ . In an ILI framework, we seek to capture knowledge of the joint distribution, $P(x, \theta)$, by building neural-network emulators for the likelihood (in red) or the posterior (in blue).

ability to specify separate proposal and prior distributions, and automated validation tests. We also include new multivariate coverage tests such as TARP (Lemos *et al.* 2023) and adaptive, automatic integration of data-loading, inference, and validation.

In this software release paper, we provide background, details, and example applications of the LTU-ILI pipeline. The code is publicly available on Github¹. In Section 2, we review the theoretical foundations of ILI. In Section 3, we describe the structure of our pipeline and the various configurations that users can interact with. In Section 4, we apply LTU-ILI on several synthetic problems where the true solution is known, to demonstrate and benchmark its performance relative to traditional methods. In Section 5, we provide numerous diverse examples of parameter inference in science using LTU-ILI. In Section 6, we discuss the important considerations to take into account when designing an ILI pipeline, as well as possible failure modes and relevant solutions. Lastly, we provide a summary of our findings in Section 7.

2. IMPLICIT LIKELIHOOD INFERENCE

The goal of ILI is the same as any Bayesian inference problem: given a model and observed data $\mathbf{x}_o$, we wish to know the distribution of model parameters $\boldsymbol{\theta}$ which are consistent with our data, *i.e.*, the *posterior distribution* $\mathcal{P}(\boldsymbol{\theta}|\mathbf{x}_o)$. The data and parameters can take many forms. In astronomy, we might consider the observed data $\mathbf{x}_o$ to be, for example, the observed X-ray photon count of a galaxy cluster, the gravitational wave signal from a collapsing black hole binary, or the full 2D sky projection of galaxy weak lensing measurements. From this data, we might wish to infer parameters $\boldsymbol{\theta}$ such as, respectively, the mass of a galaxy cluster, the mass ratio and ellipticity of the black hole binary, or the cosmological parameters governing dark matter and dark energy. For Bayesians, the posterior $\mathcal{P}(\boldsymbol{\theta}|\mathbf{x}_o)$ is the key to constraining our belief in physical laws after observing new data $\mathbf{x}_o$.

According to Bayes’ theorem, the posterior is composed of

$$\mathcal{P}(\boldsymbol{\theta}|\mathbf{x}) \propto \mathcal{P}(\mathbf{x}|\boldsymbol{\theta}) \mathcal{P}(\boldsymbol{\theta}), \quad (1)$$

where $\mathcal{P}(\mathbf{x}|\boldsymbol{\theta})$ is the *likelihood*, which defines the probability of the data given the model and some parameters, and $\mathcal{P}(\boldsymbol{\theta})$ is the *prior*, which specifies our prior belief on the true distribution of the parameters. A demonstration of the differences of the prior, posterior, and likelihood in an inference problem is given in Figure 1.

There are two approaches to modeling the terms in Equation 1: explicit (theory-driven) analyses or implicit (data-driven) analyses. In a classical *explicit* likelihood-based analysis, we assume an analytic form for the likelihood, $\mathcal{P}(\mathbf{x}|\boldsymbol{\theta})$, and substitute it into Equation 1. Typically, this expression is derived from analytic theory or simple simulations, coupled with strong assumptions on the statistical distribution of noise or unknown physics. Sometimes, these statistical assumptions are relatively simple, *e.g.*, a Gaussian distribution for continuous data (*e.g.*, Tegmark *et al.* 2004; Christensen and Meyer 2022; Porqueres *et al.* 2022) or a Poisson distribution for discrete counts (*e.g.*, Boese and Doebereiner 2001; Braun *et al.* 2008; Tsaprazi *et al.* 2023). Although these may be appropriate distributions for many situations, for a general inference problem we may not know which distribution to choose or whether this choice is valid. Also, MCMC-like samplers scale poorly with dimensionality and have bad mode coverage. This can cause problems for high-dimensional inference problems, especially if the likelihood is not differentiable with respect to the parameters. However, if we have access to a set of examples of the data-parameter relationship (*i.e.*, samples from the joint distribution, $(\mathbf{x}, \boldsymbol{\theta}) \sim \mathcal{P}(\mathbf{x}, \boldsymbol{\theta})$), we can use *implicit* likelihood analyses to learn the shape of a likelihood or posterior automatically with full differentiability. This flexibility makes ILI an extremely powerful tool for modeling complex or otherwise poorly-understood physical phenomena.

ILI learns approximate posteriors $\hat{\mathcal{P}}(\boldsymbol{\theta}|\mathbf{x})$ or likelihoods $\hat{\mathcal{P}}(\mathbf{x}|\boldsymbol{\theta})$ from the distribution of example data-parameter pairs in a training catalog, $\mathcal{D}_{\text{train}} := \{(\mathbf{x}_i, \boldsymbol{\theta}_i)\}_{i=1}^{N_{\text{train}}}$. These data can take any form; they can be sensor readings in an experiment (*e.g.*, Dingeldein *et al.* 2023), images of the night sky (*e.g.*, Ntampaka *et al.* 2019), or even compressed summaries of high-dimensional stochas-

¹ <https://github.com/maho3/ltu-ili>

tic properties (*e.g.*, Modi *et al.* 2023). They can be gathered from a pre-run simulation (*e.g.*, Fluri *et al.* 2022), be simulated on-the-fly (*e.g.*, Alsing *et al.* 2019), or curated from manually-labeled observations (*e.g.*, Lanusse *et al.* 2018), but they must be representative of the true underlying problem. Given these data-parameter pairs, we wish to construct a model capable of doing inference on real data, *i.e.*, to evaluate the posterior at some observed value $\mathbf{x} = \mathbf{x}_o$.

In this section, we describe the theoretical foundations of LtU-ILI. We begin by providing a simple introductory example for posterior inference with Approximate Bayesian Computation (ABC; Section 2.1). We then explain how one can vastly accelerate this process using neural networks to emulate probability distributions (Section 2.2). We then discuss how to practically frame machine learning in the context of Bayesian inference (Sections 2.2.1, 2.2.2 & 2.2.3). Lastly, we discuss auxiliary topics on utilizing active learning to accelerate training (Section 2.4) and validating black-box models to ensure robustness in real-world applications (Section 2.5).

2.1. Approximate Bayesian Computation

Approximate Bayesian Computation (ABC; Rubin 1984) is the original basis of ILI. Given a prior $\mathcal{P}(\boldsymbol{\theta})$, a simulator ($\boldsymbol{\theta} \rightarrow \mathbf{x}$), and an observation $\mathbf{x}_o$, ABC can construct a posterior. The general procedure of Rejection ABC is straightforward: (1) first, draw candidate parameter values from the prior $\boldsymbol{\theta} \sim \mathcal{P}(\boldsymbol{\theta})$, (2) forward-simulate these parameter values to observations $\mathbf{x} \sim \mathcal{P}(\mathbf{x}|\boldsymbol{\theta})$, (3) measure a distance metric between the candidate observations and the real observation, $d(\mathbf{x}, \mathbf{x}_o)$, and (4) if the distance is sufficiently close to the real observation (*i.e.*, $d(\mathbf{x}, \mathbf{x}_o) < \epsilon$ for small ϵ), accept the candidate parameters $\boldsymbol{\theta}$ as samples from the inferred posterior. ABC has been widely adopted in statistical inference because it does not require explicit knowledge of the likelihood and the accepted samples are guaranteed to converge to the true posterior as $\epsilon \rightarrow 0$. When ϵ is non-zero, the inferred posterior is guaranteed to be broader than the true posterior, enabling conservative inference. Despite successful applications in astrophysics and cosmology (*e.g.*, Schafer and Freeman 2012; Cameron and Pettitt 2012; Hahn *et al.* 2017), the ultimate problem with ABC methods is that the acceptance rate vanishes exponentially as the dimensionality of $\boldsymbol{\theta}$ increases, requiring thus significantly more simulations, which can be alleviated with a larger ϵ at the cost of targeting a broader posterior than the true posterior (Alsing *et al.* 2018). This limitation makes ABC highly intractable in regimes where running simulations is expensive. Occasionally ABC is used as a synonym for LFI/SBI/ILI, however throughout the paper we exclusively use ABC to refer to the algorithm discussed above.

2.2. Neural Density Estimation

In modern applications, implicit inference is made tractable by using machine learning models to emulate conditional probability distributions, a procedure called Neural Density Estimation (Papamakarios 2019). As an example, consider a situation wherein we seek to train a mixture density network (MDN; Bishop 1994) to directly model the conditional distribution $\mathcal{P}(\mathbf{v}|\mathbf{u})$ using

our training set of data-parameter pairs $(\mathbf{x}, \boldsymbol{\theta}) \sim \mathcal{D}_{\text{train}}$. For this subsection only, we have $(\mathbf{u}, \mathbf{v}) = (\mathbf{x}, \boldsymbol{\theta})$ or $(\mathbf{u}, \mathbf{v}) = (\mathbf{x}, \boldsymbol{\theta})$ interchangeably. The goal is to build a neural architecture $q_{\mathbf{w}}(\mathbf{v}|\mathbf{u})$ with weights $\mathbf{w}$ that outputs a probability distribution over $\mathbf{v}$ which targets the conditional probability $\mathcal{P}(\mathbf{v}|\mathbf{u})$. An MDN outputs probability distributions by allowing the neural network to specify the statistical parameters of a mixture density distribution. In the independent Gaussian case, this means the neural network outputs a mean and a variance for each component of the mixture density distribution, *i.e.*,

$$q_{\mathbf{w}}(\mathbf{v}|\mathbf{u}) = \frac{1}{N_c \sqrt{2\pi}} \sum_{i=1}^{N_c} \frac{1}{\sigma_i(\mathbf{u}; \mathbf{w})} e^{-\frac{(\mathbf{v} - \mu_i(\mathbf{u}; \mathbf{w}))^2}{2\sigma_i^2(\mathbf{u}; \mathbf{w})}}, \quad (2)$$

where N_c is the chosen number of components of the mixture density distribution, $\mathbf{w}$ are the learned weights and biases of the neural network, and $\mu_i(\mathbf{u}; \mathbf{w})$ and $\sigma_i(\mathbf{u}; \mathbf{w})$ are the neural network outputs representing the mean and standard deviation (or *scale*) of the i -th Gaussian component as a function of the input $\mathbf{u}$ and the weights.

To train such a network to properly emulate the conditional probability, all we need to do is maximize the joint likelihood of our training data $\mathcal{D}_{\text{train}}$. This is equivalently and often more conveniently expressed in the form of minimizing the negative log-probability loss (typically log-likelihood or log-posterior, see sections 2.2.1 to 2.2.3)

$$\mathcal{L}_{\text{MDN}} := -\mathbb{E}_{\mathcal{D}_{\text{train}}} [\log q_{\mathbf{w}}(\mathbf{v}|\mathbf{u})], \quad (3)$$

where the expectation $\mathbb{E}_{\mathcal{D}_{\text{train}}}$ is taken over all data-parameter pairs in the training set $\sim \mathcal{D}_{\text{train}}$. For the case of the the posterior distribution ($\mathbf{v} = \boldsymbol{\theta}, \mathbf{u} = \mathbf{x}$), if the training data is sufficiently diverse and if the MDN is sufficiently flexible, then the minimization of $\mathcal{L}_{\text{MDN}}$ over the network weights $\mathbf{w}$ will converge $q_{\mathbf{w}}(\boldsymbol{\theta}|\mathbf{x})$ to the true posterior $\mathcal{P}(\boldsymbol{\theta}|\mathbf{x})$ (Papamakarios and Murray 2016).

The MDN is just one example of an ever-increasing variety of neural density estimators in the literature. Another very popular class of methods is normalizing flows (Papamakarios *et al.* 2021), which output a flexible conditional distribution defined via learnable, invertible transformations of a base Gaussian distribution. Normalizing flows are often better equipped for handling cases wherein the target distribution is non-Gaussian, but low-dimensional and uni-modal. For any type of neural density estimator, the training procedure always involves minimizing the negative log-probability of training data, as in Equation 3.

Now we have established the tools for neural density estimation, we can apply them to fitting components of Bayes' theorem (Equation 1).

2.2.1. Neural Posterior Estimation (NPE)

The most straightforward method to do this is to train neural networks to directly emulate the posterior distribution, also known as Neural Posterior Estimation (NPE; Papamakarios and Murray 2016; Greenberg *et al.* 2019). As in the MDN example in the previous section, the loss function for NPE is simply the negative log-posterior of our learned neural density estimator $\hat{\mathcal{P}}(\boldsymbol{\theta}|\mathbf{x})$, and can be trained using a loss function like that of Papamakarios

and Murray (2016):

$$\begin{aligned}\mathcal{L}_{\text{NPE}} &:= -\mathbb{E}_{\mathcal{D}_{\text{train}}} \log \hat{\mathcal{P}}(\boldsymbol{\theta}_i | \mathbf{x}_i) \\ &= -\mathbb{E}_{\mathcal{D}_{\text{train}}} \log \left[\frac{p(\boldsymbol{\theta})}{\tilde{p}(\boldsymbol{\theta})} q_{\mathbf{w}}(\boldsymbol{\theta} | \mathbf{x}) \right],\end{aligned}\quad (4)$$

where our neural posterior $\hat{\mathcal{P}}(\boldsymbol{\theta}_i | \mathbf{x}_i)$ is decomposed into a neural network output $q_{\mathbf{w}}(\boldsymbol{\theta} | \mathbf{x})$ and a weighting factor taken as the ratio of the assumed prior $p(\boldsymbol{\theta})$ to the proposal prior $\tilde{p}(\boldsymbol{\theta})$. The proposal prior is defined as the distribution of $\boldsymbol{\theta}$ present in the training dataset $\mathcal{D}_{\text{train}}$, while the assumed prior is an experimental design choice representing the assumed knowledge of the global distribution of $\boldsymbol{\theta}$. In many cases, the assumed and proposal priors are identical, however this distinction is particularly relevant in the case of sequential learning (see section 2.4). Note, that the normalization factor $p(\boldsymbol{\theta})/\tilde{p}(\boldsymbol{\theta})$ cancels out when optimizing $\mathcal{L}_{\text{NPE}}$ over network weights $\mathbf{w}$, but is still required for constructing the posterior estimator $\hat{\mathcal{P}}(\boldsymbol{\theta} | \mathbf{x})$.

This method has the advantage that it allows for quick evaluation and sampling of the full posterior at inference time. This accelerated emulation of the posterior is often called *amortized* inference. However, the disadvantage is that it requires knowledge of the analytic form of the proposal prior $\tilde{p}(\boldsymbol{\theta})$ from which training data was sampled, which may not be accessible for astrophysical training data (*e.g.*, cluster properties in cosmological simulations). Recently, Vasist *et al.* (2023) used an NPE strategy to constrain exoplanetary atmospheric models. NPE has also become a known tool in Gravitational-Wave astronomy where analytical Compact Binary Coalescent (CBC) waveforms play the role of the forward model (*e.g.*, with DINGO from Dax *et al.* 2021a, and the constraints on GW150914 by Crisostomi *et al.* 2023, among many others).

2.2.2. Neural Likelihood Estimation (NLE)

An alternative method which circumvents the issue of fixed priors is to only fit for the likelihood, $\mathcal{P}(\mathbf{x} | \boldsymbol{\theta})$, via Neural Likelihood Estimation (NLE; Alsing *et al.* 2018, 2019; Papamakarios *et al.* 2019). The loss function for NLE notably does not include the assumed or proposal prior, and simply maximizes the global log-likelihood,

$$\mathcal{L}_{\text{NLE}} := -\mathbb{E}_{\mathcal{D}_{\text{train}}} \log q_{\mathbf{w}}(\mathbf{x} | \boldsymbol{\theta}), \quad (5)$$

wherein we emphasize the swapped arguments of q as compared to Equation 4. Multiplying a learned likelihood with an assumed prior results in a proxy which is proportional to the posterior, *i.e.*, $\hat{\mathcal{P}}(\boldsymbol{\theta} | \mathbf{x}) \propto q_{\mathbf{w}}(\mathbf{x} | \boldsymbol{\theta}) p(\boldsymbol{\theta})$. Generating samples from the posterior is then simply a matter of using this proxy in running Markov chain Monte Carlo (MCMC; Robert *et al.* 1999) or Variational Inference (Blei *et al.* 2017). For example, in Metropolis-Hastings MCMC (Robert *et al.* 2004), the learned likelihood and assumed prior can be used to calculate the acceptance probability of a transition $\boldsymbol{\theta}_t \rightarrow \boldsymbol{\theta}'$ without knowledge of the full normalized posterior, *i.e.*,

$$\alpha := \min \left(1, \frac{\mathcal{P}(\boldsymbol{\theta}') \mathcal{P}(\mathbf{x} | \boldsymbol{\theta}') q(\boldsymbol{\theta}' | \boldsymbol{\theta}_t)}{\mathcal{P}(\boldsymbol{\theta}_t) \mathcal{P}(\mathbf{x} | \boldsymbol{\theta}_t) q(\boldsymbol{\theta}_t | \boldsymbol{\theta}')} \right), \quad (6)$$

where $q(\boldsymbol{\theta}' | \boldsymbol{\theta}_t)$ is the proposal mechanism. Note, the process of obtaining a posterior is identical to the ex-

plicit likelihood-based approach, but instead of an analytic likelihood function, NLE has essentially a “black box” function instead. The advantage of this method is that one only needs to perform training once for a given model, at the cost of additional sampling once data becomes available. For instance, Jeffrey *et al.* (2021) fit a density model for the likelihood of compressed summaries from DES SV weak lensing maps. The Madminer tool (Brehmer *et al.* 2020) for particle physics notably allows for NLE using ILI tools.

2.2.3. Neural Ratio Estimation (NRE)

As in NLE, Neural Ratio Estimation (NRE; Hermans *et al.* 2020) targets a proxy that is proportional to the posterior. Instead of outputting a conditional density estimate for the likelihood, NRE models instead target the likelihood ratio,

$$r(\mathbf{x}, \boldsymbol{\theta}) := \frac{\mathcal{P}(\mathbf{x} | \boldsymbol{\theta})}{\mathcal{P}(\mathbf{x} | \boldsymbol{\theta}_0)}, \quad (7)$$

equal to the ratio of the likelihood at a given point in parameter space to the likelihood at a reference point, $\boldsymbol{\theta}_0$. This formulation is convenient because it can be folded into an acceptance ratio like Equation 6 and also be cast as a classification problem (Cranmer *et al.* 2015). The output of a classification network, $d_{\mathbf{w}}(\mathbf{x}, \boldsymbol{\theta}) \in [0, 1]$, can be considered equivalent to a likelihood ratio as

$$\hat{r}(\mathbf{x}, \boldsymbol{\theta}) = \frac{d_{\mathbf{w}}(\mathbf{x}, \boldsymbol{\theta})}{1 - d_{\mathbf{w}}(\mathbf{x}, \boldsymbol{\theta})}. \quad (8)$$

The classification problem can be interpreted as quantifying whether observed data $\mathbf{x}$ is consistent with $\boldsymbol{\theta}$. We can then train this neural network model using the loss:

$$\mathcal{L}_{\text{NRE}} := \mathbb{E}_{\mathcal{D}_{\text{train}}} [d_{\mathbf{w}}(\mathbf{x}, \boldsymbol{\theta}) + \mathbb{E}_{\boldsymbol{\theta}' \sim p(\boldsymbol{\theta})} [1 - d_{\mathbf{w}}(\mathbf{x}, \boldsymbol{\theta}')]], \quad (9)$$

wherein we associate a positive classification with assigning a data $\mathbf{x}$ to the correct $\boldsymbol{\theta}$, and a negative classification with those assigning data to a $\boldsymbol{\theta}'$ sampled from the prior. The main benefit of NRE over NPE and NLE is that one need not specify an approximating distribution such as MDNs or normalizing flows to emulate a posterior. The complexity is purely limited in terms of the depth and design of the neural architecture. Among others, Cole *et al.* (2022) and Karchev *et al.* (2023) applied a variation of NRE (namely Truncated Marginal NRE, Miller *et al.* 2021) to CMB data and Supernova data, respectively. Additionally, Delaunoy *et al.* (2020) and Bhardwaj *et al.* (2023) use NRE for Gravitational Wave parameter inference.

2.3. Choosing an Estimator

The optimal choice of NPE, NLE, or NRE is highly dependent on the properties on the underlying problem, the dimensionalities of $\mathbf{x}$ and $\boldsymbol{\theta}$, and the intended application. First, the ease of learning the shape of a likelihood or posterior distribution can vary dramatically between different physical problems. For example, the shape of the unimodal posterior in Figure 1 is considerably simpler than that of the bimodal likelihood, and thus NPE will be easier to converge than NLE. However, this is the opposite for some applications presented in Section 4, where the likelihood is simpler and thus

NLE converges much faster than NPE or NRE. In cases where both the posterior and likelihood exhibit complicated shapes, NRE models are a strong option, as they do not require an explicit choice of NDE.

Another strong rule of thumb is that neural networks are easier to train when using high-dimensional input to produce low-dimensional output than vice versa. Thus, if $\dim(\mathbf{x})$ is large (*e.g.*, for image- or graph-like data) then NPE may work better than NLE models. Alternatively, for high-dimensional posterior inference, *i.e.*, when $\dim(\boldsymbol{\theta})$ is large, NLE often performs better than NPE, especially when strong degeneracies exist between parameters. The NRE implementations in LTU-ILI also struggle in this regime, as evidence suggests that NRE methods require truncation or marginalization to resolve regions of high posterior density (Miller *et al.* 2021; Miller *et al.* 2022). This heuristic may change with the advent of extremely flexible diffusion architectures (*e.g.*, Song *et al.* 2020), but such models are not yet integrated into LTU-ILI.

Lastly, it is imperative to consider the downstream applications of the inference. NPE models are much better for cases in which the inference must be repeated for many tests $\mathbf{x}_o$ (*e.g.*, estimating morphology for many JWST galaxies), as it can be prohibitively expensive to run MCMC chains for NLE and NRE for many test points. However, if the learned posteriors are going to be injected into a broader hierarchical likelihood sampling (*e.g.*, using cluster masses to constrain cosmology), NLE or NRE will be a more natural choice.

For a quantitative comparison of NPE, NLE, and NRE on machine learning benchmarks, see Lueckmann *et al.* (2021). In any case, the best way to choose a method for a new problem is to try each and perform quantitative comparisons, tests which are made considerably easier with the configuration interface in LTU-ILI.

2.4. Sequential Learning

Thus far, we have phrased our problem in such a way that we first have a set of pre-run simulations, and only later do we obtain the posterior distribution given our observations. This can be the case when the simulations are particularly expensive (*e.g.*, N -body simulations in cosmology, or Earth systems models for climate science *e.g.*, from which Watson-Parris *et al.* 2021 built emulators). In these cases, we must be careful about any difference between the proposal prior for the parameters from which the simulations were run (*e.g.*, a Latin hypercube) and the chosen prior for the parameters used in the inference (*e.g.*, a multivariate Gaussian). It may be the case that this proposal prior is significantly wider than the posterior distribution inferred once the real data have been observed, meaning that many of the simulations used in training for NPE/NLE/NRE have been “wasted” as the posterior has almost no support in the region of parameter space occupied by a (potentially significant) fraction of the simulations.

One can instead adopt a multi-round inference approach to improve simulation efficiency, referred to as Sequential NPE/NLE/NRE or SNPE/SNLE/SNRE. In this method, we begin by running a fraction of the total desired simulations with parameters sampled from the broad prior $p(\boldsymbol{\theta}) = \tilde{p}(\boldsymbol{\theta})$, conducting the first round of training for the NPE/NLE/NRE model. This weak in-

ference is then applied to $\mathbf{x}_o$ to derive a first estimate for the posterior. For instance, in the case of NPE, this is equivalent to minimizing Equation 4 with $p(\boldsymbol{\theta}) = \tilde{p}(\boldsymbol{\theta})$ during the first round, which gives a model $q_w(\boldsymbol{\theta}|\mathbf{x})$, and then retraining the model with new simulations drawn from $\tilde{p}(\boldsymbol{\theta}) = q_w(\boldsymbol{\theta}|\mathbf{x} = \mathbf{x}_o)$ for the second round, and so on. The algorithm (*e.g.*, Greenberg *et al.* 2019) then identifies regions of high posterior density around the observation $\mathbf{x}_o$ and conducts additional simulations within these areas to produce a refined posterior estimate. This can be repeated until a convergence criterion is met or the total simulation budget is exhausted. This iterative refinement of the posterior around the region of interest can lead to improved posterior estimates for a given experiment. However, it is important to note that these results are deliberately optimized for a singular observed data point. The sampled simulations may not be appropriate for a different experiment, potentially resulting in a less accurate estimate of the posterior distribution if no further simulations are run.

Note that an alternative method for generating parameters of interest using acquisition functions that minimize the expected uncertainty in the posterior density approximation exists. This is based on the Bayesian Optimization for Likelihood-Free Inference (BOLFI) framework (Gutmann *et al.* 2016) has also been suggested and applied to cosmological data (Leclercq 2018), although this is not yet part of LTU-ILI.

2.5. Validation

The goal of model validation is to assess whether the posteriors $\hat{P}(\boldsymbol{\theta}|\mathbf{x}_o)$ learned in Section 2.2 will be accurate and reliable when applied to new data. Explicitly, we want to evaluate: (1) whether the learned posteriors are maximally constraining of $\boldsymbol{\theta}$ given the observed data $\mathbf{x}_o$, and (2) whether the predictive uncertainties quoted by our learned model are accurately calibrated to our training data. These two criteria are naturally adversarial, and is often referred to as the bias-variance trade-off. For example, a model can satisfy condition (1) by reporting tight error bars, but it will then fail condition (2) when its error bars are smaller than the true distribution of training data. Similarly, a model that produces a posterior equal to the prior will easily satisfy condition (2) but will ultimately be uninformative for solving condition (1). Despite the ‘black box’ nature of neural networks, these criteria control the quality of the learned implicit inference posteriors, mitigating the chance that the learned model is not representative of the data used to train it.

We evaluate the constraints and coverage of our learned posteriors on a labeled test set of data-parameter pairs $\mathcal{D}_{\text{test}} := \{(\mathbf{x}_i, \boldsymbol{\theta}_i)\}_{i=1}^{N_{\text{test}}}$. This test set is meant to represent a fully independent sampling of the real data distribution, unseen by the model prior to validation. This test set may originate from the same source as our labeled training set $\mathcal{D}_{\text{train}}$ but must be held independent to avoid overfitting and underestimation of the uncertainty (see Huppenkothen *et al.* 2023 for a review of best practices). If the distribution of data-parameter pairs in the test set matches that of the predictions of our learned posterior, then our model is calibrated and can be reliably extended to new, unlabeled data.

We build tests by comparing the true parameters from the test set to posterior samples derived from our models. Sampling-based comparisons provide a unified framework for all NPE/NLE/NRE learning strategies since they do not require normalization of the posterior PDF. The technical details for sampling from neural networks depend on the implemented learning strategy and are discussed in more detail in Section 3.3. Here, we will assume that, for a given input $\mathbf{x}$, we can draw independent and identically distributed (i.i.d.) samples from the learned posterior distributions $\hat{\boldsymbol{\theta}} \sim \hat{\mathcal{P}}(\boldsymbol{\theta}|\mathbf{x})$. These samples can then be used to compare the level of constraint on $\boldsymbol{\theta}$ and the consistency with the true values.

To test precision, which reflects the ability of a learned posterior to constrain parameter values, we examine the distribution and shape of posterior samples $\hat{\boldsymbol{\theta}}$ around the true value $\boldsymbol{\theta}$. These distributions are commonly visualized using multivariate corner plots or marginal true vs. predicted plots. The contours in these plots help qualitatively assess constraining power, identify posterior degeneracies, and detect systematic biases.

A quantitative measure of precision is the cumulative likelihood of the test dataset, denoted as $\hat{\mathcal{P}}(\mathcal{D}_{\text{test}}) = \prod_{i=1}^{N_{\text{test}}} \hat{\mathcal{P}}(\boldsymbol{\theta}_i|\mathbf{x}_i)$. Larger test likelihoods indicate a model posterior that concentrates more probability mass around the true value, implying greater constraining power. The test likelihood is simple to calculate for NPE methods, which directly estimate $\mathcal{P}(\boldsymbol{\theta}|\mathbf{x})$. However, for NLE and NRE methods, evaluating the log-likelihood involves an additional step of training a generative model $\mathcal{G}(\boldsymbol{\theta})$ from the inferred posterior samples $\hat{\boldsymbol{\theta}}$. This process is computationally expensive, especially for large datasets, and not sensitive to overconfidence.

Lastly, given samples from a reference posterior, we can evaluate the discrepancy between a model’s prediction and its optimum. Reference posteriors can often be attained via long-run MCMC chains for explicit likelihood approaches, and then used to compare with ILI methods. Lueckmann *et al.* (2021) performed an extensive quantitative analysis on various sample-wise distances for benchmarking ILI methods and found that Classifier 2 Sample Tests (C2ST; Lopez-Paz and Oquab 2016) were the most constraining measure of accuracy. C2ST is defined as the accuracy of a trained classifier (often a neural network itself) to distinguish between the true and inferred posterior samples. High C2ST suggests that the true and inferred posterior are very different, whereas a C2ST of ~ 0.5 suggests they are nearly indistinguishable. C2ST is later used to benchmark LTU-ILI in Section 4.1.

Posterior samples are also useful for quantifying the calibration of our model uncertainty in parameter space. We can construct a direct comparison of the predicted percentiles by our inference engine to the true error observed in our validation dataset. We first define the Probability Integral Transform (PIT; Cook *et al.* 2006) as the *cumulative density function* (CDF) of our model posterior given an input $\mathbf{x}_o$,

$$\text{PIT}(\boldsymbol{\theta}; \mathbf{x}_o) = \int_{-\infty}^{\boldsymbol{\theta}} d\boldsymbol{\theta} \hat{\mathcal{P}}(\boldsymbol{\theta}|\mathbf{x}_o). \quad (10)$$

Given i.i.d. samples from the posterior, we can construct

an estimator for the PIT value as

$$\hat{\text{PIT}}(\boldsymbol{\theta}; \mathbf{x}_o) = \mathbb{E}_{\hat{\boldsymbol{\theta}} \sim \hat{\mathcal{P}}(\boldsymbol{\theta}|\mathbf{x}_o)} \left[\Theta \left(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta} \right) \right], \quad (11)$$

where Θ is the Heaviside step function. Stated plainly, the PIT value counts the number of times that our posterior samples $\hat{\boldsymbol{\theta}}$ fall below the true parameter value $\boldsymbol{\theta}$. If our model posteriors are globally consistent with the truth, i.e. we match the true posterior everywhere in data-parameter space, then the distribution of PIT values evaluated on data-parameter pairs from our test set must be uniformly distributed (Zhao *et al.* 2021):

$$\text{PIT}(\boldsymbol{\theta}_i; \mathbf{x}_i) \sim U(0, 1), \quad \forall (\mathbf{x}_i, \boldsymbol{\theta}_i) \in \mathcal{D}_{\text{test}}. \quad (12)$$

As a result of this property, the distribution of PIT values is a common goodness-of-fit metric for conditional density models (e.g. Cook *et al.* 2006; Bordoloi *et al.* 2010; Tanaka *et al.* 2018). This is also known as Simulation-Based Calibration (Talts *et al.* 2018).

To better interpret errors in the PIT distribution, we can use a percentile-percentile (P-P) plot, explicitly comparing the CDF of PIT values to the CDF of a uniform random variable. Intuitively, the P-P plot measures: ‘What percentile level is my posterior model assigning to the true value, and with what frequency does this occur in the test set?’ If the predictive posterior is well-calibrated, these should be the same, *e.g.*, we should predict the true value below the 50-th percentile 50% of the time. If it is not, the shape of the P-P plot acts as a sensitive probe of global bias or over-/under-dispersion of predictive uncertainty (see Figure 1 of Zhao *et al.* 2021 for interpretations of error with P-P plots). This is an extremely practical tool for understanding and correcting biases and over-/under-dispersion in complex ILI posteriors.

As the dimensionality of $\boldsymbol{\theta}$ increases, proper coverage of the posterior distribution requires exponentially more samples. As a result, the PIT estimator (Equation 11) becomes intractably difficult to measure with low variance in the high-dimensionality regime. In this regime, we use approximate methods to bound the constraints of posterior coverage tests. Firstly, we can construct a PIT value from the marginal posterior over each i -th component of $\boldsymbol{\theta}$, *i.e.*,

$$\hat{\text{PIT}}(\theta_i; \mathbf{x}) = \mathbb{E}_{\hat{\boldsymbol{\theta}} \sim \hat{\mathcal{P}}(\boldsymbol{\theta}|\mathbf{x})} \left[\Theta \left(\hat{\theta}_i - \theta_i \right) \right], \quad (13)$$

where $\hat{\theta}_i$ and θ_i are the i -th components of $\hat{\boldsymbol{\theta}}$ and $\boldsymbol{\theta}$ respectively. If our model posterior is globally consistent with the true posterior, then this marginal PIT value has the same properties as Equation 11, *i.e.*, when evaluated on the test set they should be uniformly distributed.

It is also sufficient to use approximations to check multivariate posterior coverage. The Tests of Accuracy with Random Points (TARP; Lemos *et al.* 2023) method uses samples in high-dimensional parameter space to estimate expected coverage probabilities. By examining the density of posterior samples within regions of parameter space near true values, TARP constructs estimates of posterior coverage which are guaranteed to converge to the true posterior coverage with sufficient samples. With enough test samples, the TARP method is necessary and sufficient to test the accuracy of posterior estimators.

3. CODE STRUCTURE

The LTU-ILI software is a pipeline for training implicit inference models to infer scalar parameters from observational data. A typical inference pipeline consists of three main stages: Data, Inference, and Validation. The Data stage involves loading datasets from file or setting up on-the-fly simulators (e.g. for the methodologies in Section 2.4) in order to provide data-parameter pairs for implicit inference. The Inference stage then takes these data-parameter pairs, trains neural networks to connect them with posterior or likelihood representations (as in Section 2.2), and saves the fitted models to file. Finally, the Validation stage loads a test dataset as well as the fitted models from Inference in order to evaluate goodness-of-fit metrics (e.g. constraint level, posterior coverage, etc.) and produces inference on unlabeled data. A schematic of this pipeline is shown in Figure 2.

Each stage is independent and adaptive, such that the configuration of one stage can be changed while keeping the others the same and the pipeline will still run. For example, if you had a fixed dataset but wanted to try out many different neural architectures, you would only need to change the Inference configuration. Similarly, if you wanted to see the gain in constraining power of using a short, conservative data vector versus a longer, more informative data vector, you could simply change the Data configuration to load the different datasets and the Inference and Validation would proceed exactly the same way. This design principle is extremely practical for doing apples-to-apples comparisons of the many design choices and hyperparameters associated with ILI, a key tenet of robust and responsible machine learning (Huppenkothen *et al.* 2023).

The pipeline is designed to promote exploratory research while also supporting efficient production-level testing. We provide a fully functional Jupyter interface and detailed examples to access each stage of LTU-ILI. The Jupyter interface is a widely-used tool for examining new problems, building educational material, and facilitating collaborative efforts (Wang *et al.* 2019). The Jupyter interface of LTU-ILI provides step-by-step guidance for establishing and testing a new inference pipeline that is familiar to novices and experts alike. This allows LTU-ILI to act both as a user-friendly toolkit to explore inference on a new problem as well as carry out thorough testing and hyperparameter tuning through its `yaml`-based configuration API. Users can generate many configurations of the pipeline and run multiple fittings in parallel on large-scale GPU compute clusters.

In this section, we discuss the design of the Data, Inference, and Validation stages of the LTU-ILI pipeline, including each of their capabilities and relevant reference works. For specifics on using the API, we refer the reader to our full code documentation².

3.1. Data

The objective of the Data stage is to gather data-parameter pairs and present them in an amenable format to the Inference stage. For standard inference, with fixed training and test catalogs, the Data stage simply loads data-parameter pairs from in-memory

or on-disk storage and provides `get_all_data()` and `get_all_parameters()` helper functions to pass along data and parameters in the correct format. These functions are called during the Inference stage for later pre-processing within the `sbi`, `pydelfi`, and `lampe` backends. Within LTU-ILI, we provide several convenience classes to prepare data from either `numpy`-, `xarray`-, or `torch`-like data formats. Users are also can create their own data-loading objects, so long as they adhere to the aforementioned conventions.

For multi-round inference (Section 2.4), the Data stage also provides a `simulate()` function, wherein the loader will execute an on-the-fly simulation for given parameters to generate data for efficient sequential learning. In the example class provided in LTU-ILI, users can initialize this stage with an arbitrary simulation function that intakes parameters θ and outputs data x , and the sequential Inference stage will handle any calls to the function. Also, users can load a pre-run initial catalog from file and write any newly-generated simulations to disk during the training procedure.

3.2. Inference

The LTU-ILI code encompasses a broad diversity of methodologies and architectures for performing implicit inference. However, the core configurations for all types of inference are universal. Every Inference stage requires specifying: a posterior-, likelihood-, or ratio-estimation; one or multiple neural architectures; a prior distribution; and training hyperparameters (e.g. learning rate, batch size, etc.).

Once these are provided, the Inference stage knows how to pull data-parameter pairs from the Data stage, perform data preprocessing, construct each training algorithm, minimize losses (see Equations 4, 5, and 9), and produce a learned posterior model, $\hat{P}(\theta|x)$. Each inference algorithm automatically implements best practices for training, including data normalization (Singh and Singh 2020), adaptive stochastic gradient descent (Kingma and Ba 2014), and validation-guided early stopping (Bai *et al.* 2021). The remainder of this section is dedicated to the core considerations of configuring the Inference stage.

The Inference stage is built on three distinct computational backends: `pydelfi` (Alsing *et al.* 2019), `sbi` (Tejero-Cantero *et al.* 2020), and `lampe` (Rozet *et al.* 2021). `pydelfi` is a Tensorflow package (Abadi *et al.* 2016) for performing NLE using active learning and data compression techniques. `sbi` is a PyTorch package (Paszke *et al.* 2019) for performing NPE, NLE, and NRE, as well as their sequential analogs. `lampe` is also a PyTorch package focused on NPE methods, with substantial support for diverse embedding networks and normalizing flow models. The implementations of NLE in `pydelfi` and `sbi` are similar in theory (Papamakarios *et al.* 2019) but differ in applications of embedding networks. `pydelfi` has the option to train in a two-step process, wherein first it trains an embedding network to optimally compress data via Fisher information maximization (Charnock *et al.* 2018; Alsing and Wandelt 2018) and then attaches it to the head of an NLE model, whereas `sbi` exclusively uses NLE embedding networks to compress parameters. `pydelfi`'s two-step

² <https://ltu-ili.readthedocs.io>

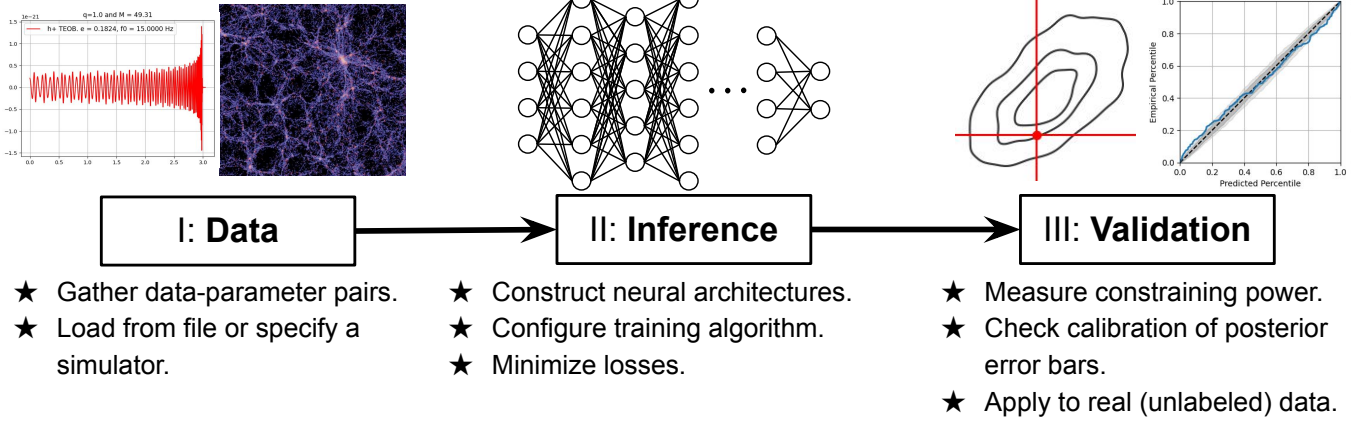


FIG. 2.— Overview of the LTU-ILI pipeline, displaying the procedural processes when running each stage. Each of the three stages (Data, Inference, Validation) is independently configurable, meaning any setup of one stage will automatically link to the others.

compression-estimation process has been shown to significantly improve simulation-efficiency during sequential training for applications in cosmology (Alsing *et al.* 2018). The `lampe` NPE in LTU-ILI is a custom implementation of Lueckmann *et al.* (2017), similar to SNPE-B of `sbi` but allowing for greater variety of configuration, embedding networks, and NDEs. LTU-ILI is a unifying framework for all of these mutually-exclusive backends, allowing, for the first time, convenient apples-to-apples comparison of each implementation of ILI.

The Tensorflow and PyTorch backends have a variety of pre-implemented NDEs and ratio classifiers. The NDEs are either Mixture Density Networks (MDNs; See Section 2.2) or normalizing flows (See Papamakarios *et al.* 2021, for a review). The `pydelfi` backend possesses its own neural architectures as originally implemented from scratch in Alsing *et al.* (2019), whereas the `sbi` and `lampe` models have software dependencies, respectively `nflows` (Durkan *et al.* 2020) and `zuko` (Rozet *et al.* 2022). Both backends also accept various standard and customizable prior distributions. The primary requirement of a custom prior is the ability to evaluate $\log p(\theta)$ (for NDE models) or a proportional proxy (for NLE/NRE models). If the assumed prior’s support is bounded (*e.g.*, a top-hat prior), the code internally implements an affine parameter transformation to avoid assigning a nonzero probability to regions of parameter space beyond prior bounds.

A key component of LTU-ILI is model ensembling, wherein we independently train multiple neural networks on the same dataset and combine their predictions to improve robustness to overfitting and model uncertainty. Singular NDEs are prone to overconfidence (Hermans *et al.* 2022), *i.e.*, they have a tendency to underestimate the predictive uncertainty. Deep ensembling (Lakshminarayanan *et al.* 2017) is a well-tested schema for correcting overconfidence by averaging NDE predictions from multiple models to inflate error bars and correctly capture model uncertainty. It is an alternative to the computationally-expensive Bayesian neural networks (Blundell *et al.* 2015; Gal and Ghahramani 2016; Cobb and Jalaian 2021), in which the full weight posterior is learned during training and sampled during inference. We recommend that users utilize deep ensembles to ensure robust inference within LTU-ILI.

The adaptability of neural networks to various data

inputs has greatly contributed to their growth in astronomy and cosmology. The convenient implementations of the various NDEs and classifiers in LTU-ILI can be further augmented with customizable embedding networks. These embedding networks can use classic `Keras`-like neural layers (Chollet *et al.* 2015) in both Tensorflow and PyTorch, greatly simplifying implementation. The integration with embedding networks in the `lampe` backend allow for exotic inputs, like graphs and sequences. In the code, we provide several examples of embedding networks, including connections to convolutional and graph neural networks (LeCun *et al.* 1998; Kipf and Welling 2016). We also note that all backends can be run on GPUs, to take advantage of computational speed-ups when using complex embedding networks.

3.3. Validation

The Validation stage provides modular functions to implement the validation metrics described in Section 2.5 on the learned posteriors $\hat{\mathcal{P}}(\theta|\mathbf{x})$. The execution of the Validation stage follows:

1. Load the learned neural posteriors from the Inference Stage.
2. Sample the posterior $\hat{\theta} \sim \hat{\mathcal{P}}(\theta|\mathbf{x})$ at test data points, $\mathcal{D}_{\text{test}}$, or at an unlabeled observational point, $\mathbf{x}_{\text{obs}}$.
3. Construct the validation metric and return or save it to file.

The most impactful configuration of the Validation stage is the choice of sampling methods. LTU-ILI provides three classes of samplers: MCMC, variational inference (VI), and direct sampling. All models (NPE/NLE/NRE) in both backends are natively integrated with the `emcee` (Foreman-Mackey *et al.* 2013) sampling package. `emcee` implements the Goodman and Weare (2010) MCMC sampler and improves efficiency through parallelization and affine transformations but does not take advantage of probability gradients. The `sbi` models also have access to PyTorch’s `pyro` samplers (Bingham *et al.* 2019), including slice samplers (Neal 2003), Hamiltonian Monte Carlo (HMC; Neal *et al.* 2011), and the No-U-Turn sampler (NUTS; Hoffman

et al. 2014). All **emcee** and **pyro** samplers have a unified interface within LTU-ILI, wherein users can simply specify their choice of sampler as well as the number, length, and thinning of MCMC chains. Also, we provide the functionality for **sbi** models to fit and sample from NDEs through neural VI (Graves 2011). VI is a fast, approximate alternative to MCMC sampling, particularly useful in the high-dimensional parameter regime. Lastly, solely for the NPE models in **sbi**, users can utilize direct sampling, which is faster than both MCMC and VI. We remark that the choice of sampling configuration might change according to the validation metric in question, so LTU-ILI allows for the flexibility to specify this on a case-by-case basis.

3.4. Choosing a Backend

As described above and later tested in Section 4, the algorithmic behavior of the **sbi**, **pydelfi**, and **lampe** backends are nearly identical for the same experimental configurations (*i.e.*, the same training data, hyperparameters, neural architectures, etc.). However, each backend has unique strengths and weaknesses and may be more appropriate for different problems. For example, **sbi** is the most commonly-used backend, with a set of standardized training procedures and architectures which have been validated across many problems (Lueckmann *et al.* 2021). It also has custom samplers for NLE and NRE methods which are faster than the generic **emcee** samplers used in the **pydelfi** backend. For new users, **sbi** is a reliable choice for exploring new datasets or setting up standard ILI applications. In contrast, **lampe**’s routines have access to a broad scope of exotic NDEs and allow for much greater flexibility when designing training procedures and custom embedding architectures. In our experience, using the flow models unique to **lampe** often produced tighter and better-calibrated posterior estimates relative to similar **sbi** applications. In addition, graph and recurrent embedding networks are currently only possible through the **lampe** interface, because of its capacity to handle complex data modalities. As a result, **lampe** is recommended for more advanced use of ILI in complex datasets. Lastly, our **pydelfi** tests in Section 4.2 appear to indicate stronger performance in sequential learning tasks than similar approaches with **sbi**. Also, **pydelfi**’s **Tensorflow** implementation may make it a more attractive option to users already working in **Tensorflow**, rather than the other options in **PyTorch**. Despite these rules of thumb, LTU-ILI is a fully unified interface incorporating all these backends. Our code allows users to easily explore the strengths of each backend for their specific applications.

4. SYNTHETIC EXPERIMENTS

In this section, we demonstrate the capabilities of LTU-ILI on a number of synthetic experiments for which an analytic form for the posterior or likelihood is known. We use these traditional experiments to measure benchmark performance the variety of inference engines available in the code against each other and classical methods (*e.g.*, ABC, HMC).

4.1. Toy Problem

We first build a toy model to demonstrate LTU-ILI’s ability to learn non-linear posteriors. We consider a 10-

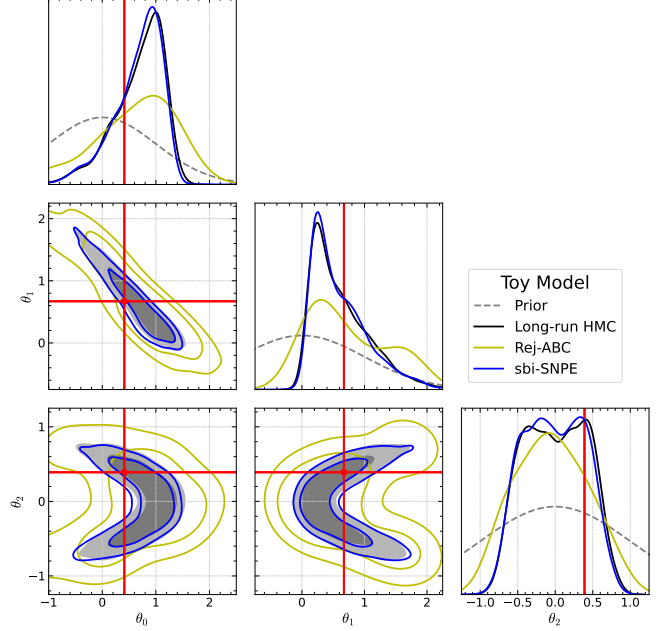


FIG. 3.— Example SNPE posterior inference in comparison to classical implicit (Rejection ABC) and explicit (HMC) likelihood inference for the known toy simulator in Equation 14. The true value for each parameter is shown in red and contours are shown at the central 68% and 95% confidence intervals. Note that the prior for each θ_i is a standard normal, as $p(\theta_i) = \mathcal{N}(0, 1)$.

dimensional data vector $\mathbf{x}$ constructed element-wise from the following stochastic simulator:

$$x_i = 3 \sin(k_i + \phi_0) + \phi_1 k_i^2 + \epsilon_i, \quad (14)$$

where $k_i = (2i/3) - 3$ and $i \in [0, 9]$. Here, the data vector $\mathbf{x} := \{x_i\}_{i=0}^9 \in \mathbb{R}^{10}$ is composed of a sinusoidal signal with phase ϕ_0 , a quadratic signal of amplitude ϕ_1 , and an i.i.d. noise component $\epsilon_i \sim \mathcal{N}(0, 1)$. We further decompose ϕ into our target parameters $\theta \in \mathbb{R}^3$ via $\phi_0 = \theta_0 + \theta_1$ and $\phi_1 = \theta_1 - 3\theta_2^2$. We expect our inference engine to forward constraints on latent variables ϕ onto target variables θ . We assume a standard normal prior on each of our parameters of interest, $p(\theta_i) = \mathcal{N}(0, 1)$. This is a challenging inference task because the likelihood is highly non-linear and the posteriors exhibit strong degeneracies. However, learning parameters from such complex, correlated data vectors is a common problem in astronomy and cosmology (*e.g.*, Dax *et al.* 2021a; Modi *et al.* 2023; Bartlett *et al.* 2023).

Given this simulator, we train **sbi**-SNPE using an ensemble of two normalizing flows (MAFs) to produce the example θ posterior in Figure 3, shown relative to equivalent constraints with Rejection ABC and long-run HMC. We train the SNPE sequentially using 10 rounds of 2000 simulations each, following the procedure described in Section 2.4 and using Equation 14. For ABC, we run the same number of simulations (20,000) and take the closest 0.5% to our data vector, using an L2 distance. For HMC, we run eight chains each with 10,000 tuning steps and 10,000 sampling steps, for a total of 160,000 simulations. This run has an effective sample size of 3,900 – 8,000 and a Gelman-Rubin $\hat{R}$ statistic of 1.001 for all parameters. We consider these long-run HMC samples to be equivalent to a reference ‘ground truth’ posterior.

Despite the same simulation budget as ABC, the SNPE

posterior constraints have converged to the reference posterior and are strongly more constraining than the ABC constraints. In both the SNPE and HMC posteriors, we clearly observe the expected linear degeneracy between θ_0 and θ_1 as well as the quadratic degeneracy between θ_1 and θ_2 , direct results from the aforementioned definitions of ϕ_0 and ϕ_1 . These two degeneracies imply another quadratic degeneracy between θ_0 and θ_2 . Despite this non-linearity, we see the multi-dimensional 68% and 95% confidence intervals are entirely consistent between SNPE and HMC. However, ABC methods struggle in this regime. Because the data dimensionality is ten, ABC requires orders-of-magnitude more samples to fully capture the tails of the posterior distribution. This regime demonstrates the utility of having a low-cost amortized posterior model like SNPE, especially in further cases where the simulators are not as simple.

4.2. Benchmarking

We benchmark all models in LTU-ILI on the standard “Simple Likelihood Complex Posterior” problem (SLCP; Papamakarios *et al.* 2019) common in simulation-based inference experiments (*e.g.*, Greenberg *et al.* 2019; Lueckmann *et al.* 2021; Ramesh *et al.* 2022). SLCP has a fairly straightforward, unimodal likelihood but a very complex, multimodal posterior distribution, making it a difficult, but attainable benchmark for implicit inference. We follow the implementation of SLCP in Lueckmann *et al.* (2021), wherein we train various NPE/NLE/NRE methods and compare their resultant posteriors to reference posteriors determined with long-run HMC chains. We implement this for all available NPE/NLE/NRE methods and their sequential analogs in each of the `pydelfi`, `sbi`, and `lampe` backends. We also include benchmarking of Rejection ABC as a baseline. Throughout all backends and engines, we use the same architectures as in Lueckmann *et al.* (2021), which were determined through exhaustive hyperparameter searches.

For each trial, we measure the error of an inference engine with respect to a reference posterior using C2ST (see Section 2.5). For each backend, engine, and simulation budget, we perform independent training and inference on ten separate reference posteriors to aggregate test statistics.

In Figure 4, we show the performance of each LTU-ILI engine on the SLCP benchmark as a function of the simulation budget used for generating the training set. As we would expect, the accuracy of inferred posteriors increases with the number of given simulations across all models. We observe that all models strongly outperform the Rejection ABC baseline due to their efficient use of simulation data. For the SLCP benchmark, we observe that NLE methods perform the best, NPE methods are a close second, and NRE methods have the worst performance. However, we note that the SLCP by definition has a simple, easy-to-learn likelihood, encouraging strong NLE performance. This hierarchy of performance could be entirely different for other problems. Sequential methods (SNPE/SNLE/SNRE) show very little improvement for small simulation budgets (10^3), but show massive gains for large budgets (10^5), especially for the NLE methods. This is likely because, for small training datasets, the posteriors are poorly constrained and

barely deviate from the prior. However, for the tight posterior constraints in large budget runs, sequential methods are able to more easily focus simulation resources on small regions of parameter space. Next, we note that the `lampe`-NPE implementation largely mirrors that of `sbi`-NPE, and `pydelfi`-NLE slightly outperforms `sbi`-NLE for a medium simulation budget (10^4). Although hyperparameters (*i.e.*, architectures, learning rate, validation schema) are fixed for these experiments, slight differences in backend implementations might lead to better or worse performance. Lastly, as a check, the performances of all `sbi` engines closely match those found in Lueckmann *et al.* (2021), suggesting our implementation of SLCP is correct.

5. SCIENCE EXPERIMENTS

In this section, we demonstrate various applications of LTU-ILI for common astrophysics and cosmology problems. These experiments are designed to display the diverse functionality of the code, including using CNN embedding networks (Section 5.1), applying posterior coverage validation (Section 5.2), incorporating graph neural network layers (Section 5.3), using multiround inference (Section 5.4), quantifying information in different data sources (Section 5.5), and integrating information-optimal embeddings (Section 5.6). We note that these experiments are intended to illuminate interesting pathways for the scientific application of LTU-ILI and not necessarily make significant scientific conclusions on their subject material. For readability, experiment descriptions are kept concise, but the code and data are made publicly available through the codebase.

5.1. X-ray Mass Estimation of Galaxy Clusters

A prime functionality of LTU-ILI is its flexibility to adapt to many modalities of data. Through the use of a customizable embedding network, we process image data to estimate the mass of galaxy clusters from X-ray observations. Our inference problem mirrors that of Ho *et al.* (2023), wherein we try to estimate the M_{500c} mass of a galaxy cluster from 128×128 images of sky-projected X-ray photon counts. To train and test our model, we use a dataset of 3,285 mock X-ray observations of clusters derived from the Magneticum hydrodynamical simulation (Dolag *et al.* 2016) designed to mimic those of the eROSITA telescope (Soltis *et al.* 2022). These mock observations are single-band images of the X-ray photon emission from the hot intra-cluster medium (ICM) and include realistic simulation of the contaminating systematics which afflict real X-ray measurements, including cluster morphology, background emission, telescope response, and AGN sources. Examples of these observables are shown in Figure 5. A good estimator of cluster mass needs to be able to disentangle the source X-ray signal from the noise and peripheral emission, while also understanding the physical connection between X-ray emission, ICM gas content, and system mass.

To recreate this example, we perform NPE using an embedding network of the same convolutional architecture as Ho *et al.* (2023). That is, instead of the last dense layer mapping to a single point estimate, we forward the final embedding into an NDE. We construct an ensemble of four models, each with the same embedding architecture, but two with Masked Autoregressive Flow (MAF;

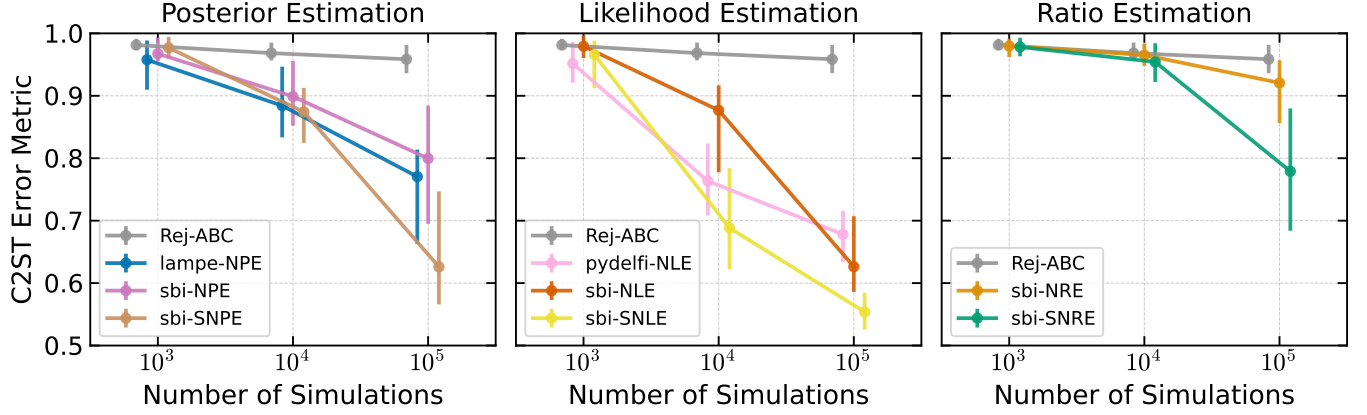


FIG. 4.— Prediction error as a function of simulation budget for various LTU-ILI training methodologies on the benchmark SLCP inference problem (Papamakarios *et al.* 2019). For clarity, plotted points for different models at the same simulation budget are slightly offset horizontally. The Rejection-ABC method is shown on all subplots as a baseline. Error is defined in terms of the Classifier 2-sample Test (C2ST; Lopez-Paz and Oquab 2016) metric. A lower C2ST indicates a more accurate inferred posterior, with 0.5 being the optimal score. C2ST values are shown at their median and central 95% confidence interval, calculated over ten independent runs.

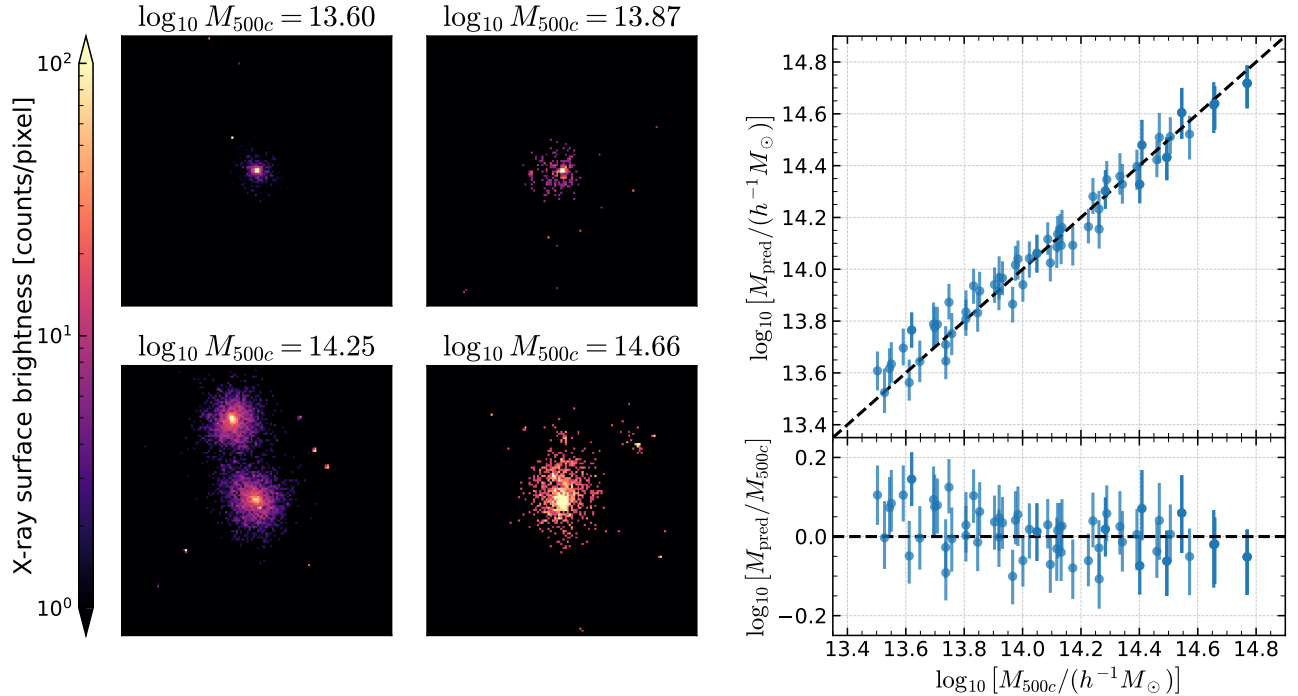


FIG. 5.— Inference of galaxy cluster M_{500c} mass from eROSITA-like mock X-ray observations, as in Ho *et al.* (2023). The left four subplots show random examples of single-band X-ray observables in terms of their sky-projected surface brightness and target mass (in $h^{-1} M_{\odot}$). The right plot is the true vs. predicted log-mass estimates on the test set, where we show the median and central 68% confidence interval of our neural posteriors. For readability, test points in the right plot have been randomly subsampled.

Papamakarios *et al.* 2017) NDEs and two with MDNs. We then train the models from scratch using the NDE losses, instead of the mean-squared-error (MSE) loss of Ho *et al.* (2023). As in the reference, we assume a uniform prior on cluster mass. We split our mock catalog into 90% training data and 10% test data, and present our performance results on the latter. The entire training and testing procedure takes about fifteen minutes on an Nvidia V100 GPU.

The true vs. predicted mass estimates for this model are shown in Figure 5. Using the original architecture, we achieve a test scatter on the mean prediction of 0.0782 dex. When compared to the 0.0773 dex scat-

ter of the single-band X-ray images in Ho *et al.* (2023), we achieve a very similar level of information extraction, though our uncertainty is slightly higher than the original. We attribute this to the fact that model ensembling slightly inflates predictive uncertainty (Hermans *et al.* 2022). However, we remark that by slightly increasing the number of convolutional filters in the first two layers, we observed mean scatters as low as 0.071 dex, suggesting that the NPEs are capable of state-of-the-art constraints on physical problems when given the right architecture. We also note that hyperparameter search through neural architectures is made efficient and accessible through LTU-ILI’s configuration-based interface.

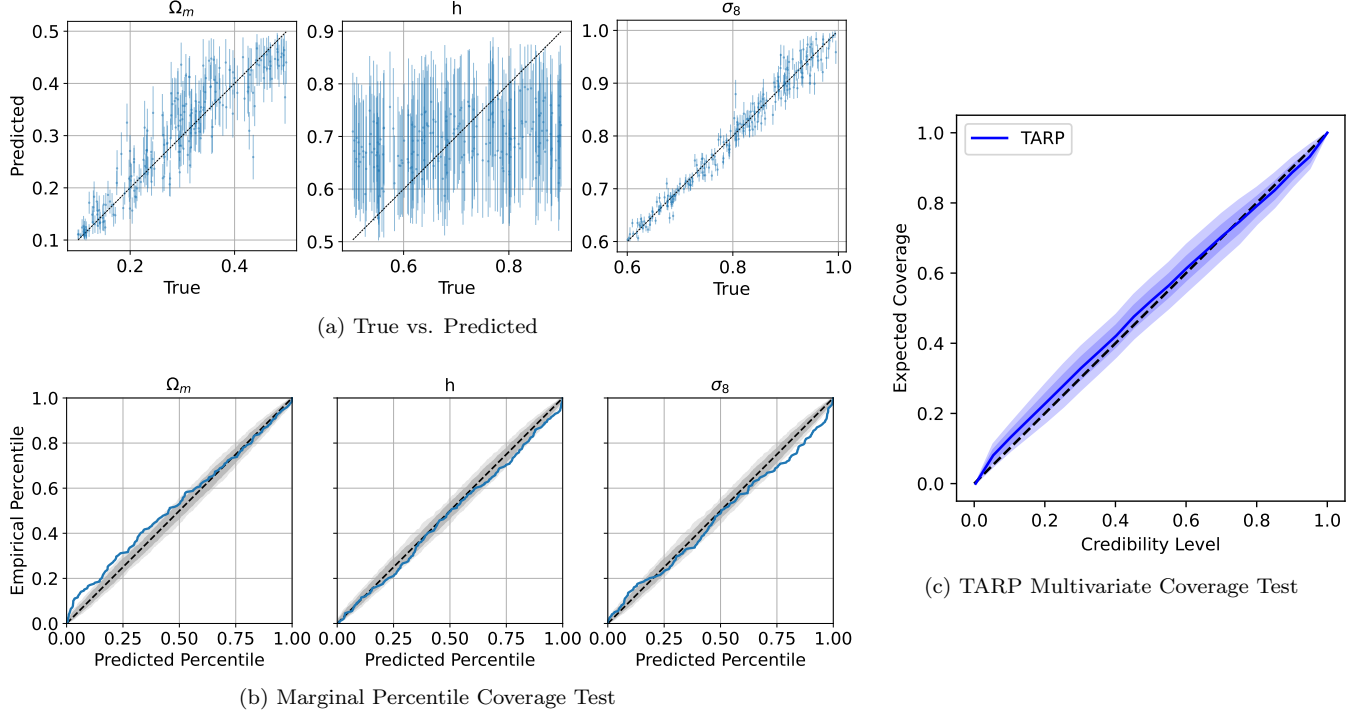


FIG. 6.— Inference of three cosmological parameters Ω_m , h , and σ_8 from halo power spectrum multipoles in Quijote simulations using LTU-ILI. In (a) we show the comparison between true and predicted value of the parameters in the test simulations. Marginal error bars are shown at the central 68% confidence intervals. In (b) and (c) we show the coverage tests using the P-P plots as well as using TARP method. These show bootstrapping error bars at 68% and 95%, indicative of typical variations in an ideal estimator for a finite test set.

5.2. Quijote Dark Matter Power Spectrum

Another prime utility of LTU-ILI is its capacity to seamlessly integrate ILI training procedures with sampling and validation metrics. A common inference benchmark in the field of cosmology is inferring cosmological parameters from the matter power spectrum (*e.g.*, Cooray and Sheth 2002; Spurio Mancini *et al.* 2022; Hahn *et al.* 2023b). In this problem, we run simulations of the matter evolution in the universe for various cosmological models, measure observational summary statistics such as the matter power spectrum, and then try to connect these observations back to constraints on cosmology. Here, we demonstrate this application by predicting the posterior distribution of three cosmological parameters using the power spectrum multipoles measured from the halo catalog of Quijote simulation (see Villaescusa-Navarro *et al.* 2020, for a description of the simulations). We use the power spectrum multipoles corresponding to $\ell = 0, 1, 2$ in 23 linearly spaced wavenumber (k) bins ranging from $0.08 h^{-1}\text{Mpc}$ to $2.8 h^{-1}\text{Mpc}$ as our observations, x . We use the NLE method to learn the likelihood $p(x|\theta)$, where the parameters θ correspond to three cosmological parameters: the total matter density Ω_m , amplitude of the matter fluctuations σ_8 , and the Hubble constant h measuring the rate of the expansion of the Universe.

Using the `sbi` backend, we train a NLE model using an ensemble of six NDEs, each using a thin MAF architecture, with ten hidden layers and three transformations. We use 1800 simulations from Quijote to train the network, retaining the remaining 200 to test its performance as shown in Figure 6. We then use VI sampling to obtain the posterior of the parameters from this trained likeli-

hood ensemble, weighted by their validation losses. This greatly accelerates the sampling process on the full test set when compared to traditional MCMC sampling.

We show the true vs predicted value of the three parameters on the top-left panels (Figure 6a), finding good predictive performance for Ω_m and σ_8 . The lack of predictability for h is expected physically, as the power spectrum multipoles alone are significantly less sensitive to expansion rate without some external prior information. We also perform coverage tests on the posteriors, finding uniformly distributed rank histograms as given by the coverage test in the bottom-left panels (Figure 6b). Finally, we show the coverage test using the TARP method in the center-right panel (Figure 6c), finding that the posterior is correctly calibrated.

5.3. Field-level Inference with Graph Neural Networks

We next demonstrate LTU-ILI’s ability to handle complex data modalities by addressing the same cosmological inference problem using a point cloud dataset. Instead of power spectrum summaries, we instead use discrete catalogs of the 10,000 heaviest dark matter halos in each of the Quijote simulations (Villaescusa-Navarro *et al.* 2020). The high resolution information present in the discrete halo positions has been shown to constrain cosmological parameters to extremely high precision (*e.g.*, Makinen *et al.* 2022b; Cuesta-Lazaro and Mishra-Sharma 2023; de Santi *et al.* 2023b). We ingest this catalog into our neural architectures using a message-passing graph neural network (Gilmer *et al.* 2020), which treats dark matter halos as nodes on a graph and filters down field-level information into highly-informative neural representations. This graph information is then fed into flow-based NDEs for performing NPE. We note that usage

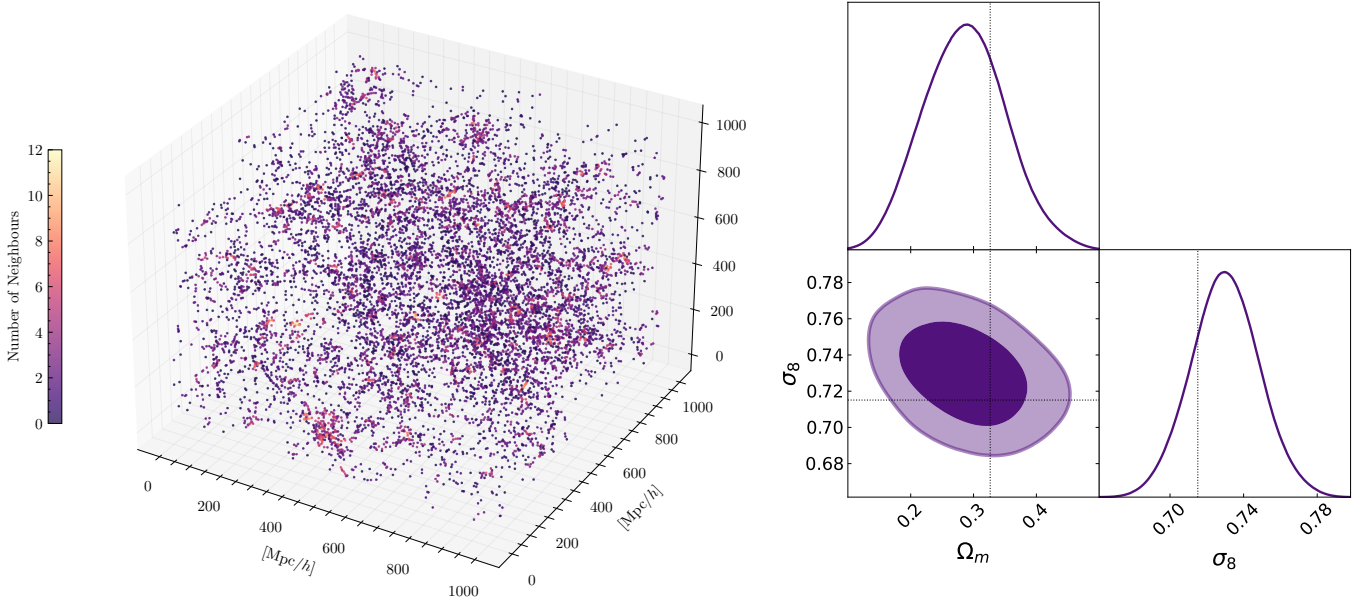


FIG. 7.— Cosmological inference from dark matter halo point clouds. Left: Example point cloud from the Quijote simulations. The graph neural network takes as an input the distances between halos separated by less than 20 Mpc/h. Right: Inferred posterior over Ω_m and σ_8 on a test set simulation. True values are shown in gray. Contours are shown at the central 68% and 95% confidence intervals.

of graph-type inputs is only possible through the `lampe` backend in LTU-ILI, as was employed here.

We train this graph-enhanced NPE model to recover the cosmological parameters Ω_m and σ_8 from the positions of Quijote’s heaviest dark matter halos. The graph neural network is a message-passing neural network with node and edge neural networks being two multi-layer perceptrons with 3 layers of 128 hidden units each. The network takes as inputs 3D distances between halos, and their corresponding modulus. Nearby halos are connected within the graph if their comoving distance smaller than 20 h^{-1} Mpc. For NDEs, we train an ensemble of four networks, each one using a MAF architecture with five transformations and 50 hidden features.

In Figure 7, we show an example point cloud together with the resulting graph neural network inference. The graph neural network obtains informative posteriors, consistent with true values. We note that the constraints are not as tight as those produced for the Quijote dark matter power spectrum in Figure 6. This is likely because the graph only targets the most massive halos, *i.e.*, a more physically realistic probe, while the results in Figure 6 access the full, high-resolution dark matter distribution. In any case, LTU-ILI with the `lampe` backend provides an easily accessible pathway for new users to integrate exotic architectures such as graph architectures into their development pipeline.

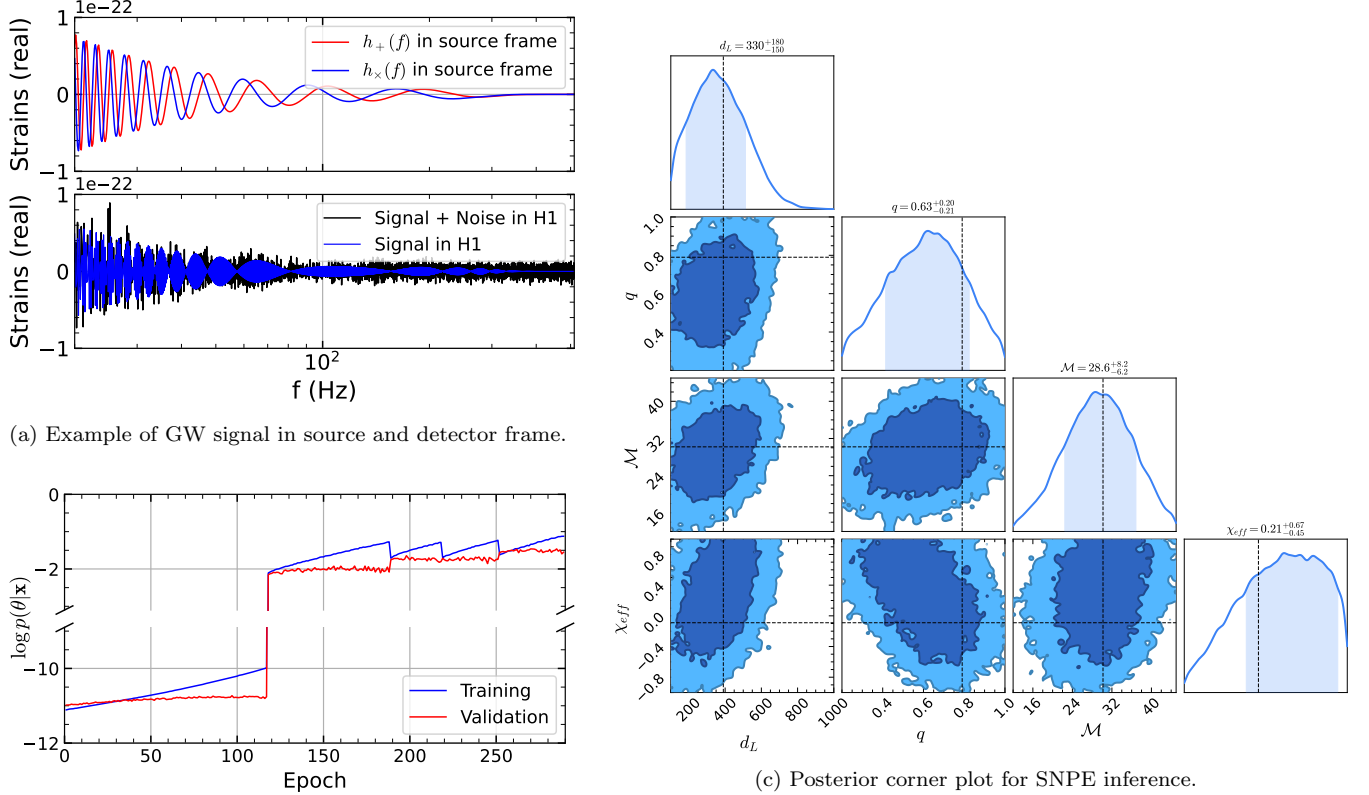
5.4. Multi-round Inference for Gravitational Waves

We next test the performance of LTU-ILI when addressing sequential learning tasks, specifically focusing on gravitational wave reconstruction. Over recent years, several studies based on ILI methods have addressed the problem of reconstructing parameters of merger events from gravitational wave (GW) signals (Cutler and Flanagan 1994; Thrane and Talbot 2019). Initial advancements were marked by significant contributions to poste-

rior estimation using normalizing flows, as highlighted in Green *et al.* (2020); Green and Gair (2020). Subsequent research (DINGO: Dax *et al.* 2021a) introduced variations of NPE taking advantage of Group Equivariant Neural Posterior Estimation (GNPE: Dax *et al.* 2021b) or Flow Matching Posterior Estimation (Wildberger *et al.* 2023). More recently, Crisostomi *et al.* (2023) showcased a similar application of multi-round NPE to estimate ringdown parameters of GW150914. Delaunoy *et al.* (2020) and Bhardwaj *et al.* (2023) also serve as recent illustrations of applying NRE to GW analyses – the latter incorporating multi-round inference as well.

In this example, we explore the application of multi-round inference to constrain a reduced set of gravitational wave (GW) parameters. Our forward-modeled data vector is constructed by merging the simulated frequency-domain strain signals from each of the Hanford (H1) and Livingston (L1) detectors. We employ the IMRPhenomPv2 algorithm (Khan *et al.* 2016) with a frequency resolution of $\Delta f = 0.125$ Hz, resulting in an input data vector of $\dim(\mathbf{x}) = 2 \times 7872$. An illustration of this data vector is presented in Figure 8a for clarity. We stress that, in contrast to likelihood-based sampling methods implemented in `Bilby` (Ashton *et al.* 2019; Romero-Shaw *et al.* 2020), which only necessitate the signal component of the forward model for the likelihood computation, our ILI approach requires the inclusion of noise realizations in the training data vector.

We illustrate a simplified application of LTU-ILI for SNPE on GW signals, fixing parameters related to the time and geometry of the detection and focusing on a set of four parameters: chirp mass ($\mathcal{M}$), mass ratio (q), effective aligned spin (χ_{eff}) and luminosity distance (d_L). These parameters, chosen for their reduced correlation and significance in GW parameter estimation (*e.g.*, Veitch *et al.* 2015; Vitale *et al.* 2017), are derived from the masses of the coalescing objects – with



(a) Example of GW signal in source and detector frame.

(b) Multi-round training and validation posterior probability.

(c) Posterior corner plot for SNPE inference.

FIG. 8.— Example parameter inference of a GW150914-like event as observed by the gravitational wave Hanford (H1) and Livingston detectors. (a) Example simulated gravitational signal. Top: Radiation polarizations h_+ and $h_\times$ in source frame. Bottom: Real part of the strain and noise realization on Hanford (H1) detector frame. (b) Evolution of the log-probability for the multi-round GW inference example using SNPE, *i.e.*, the negative of the loss function in equation (4) for $n_r = 5$ rounds. (c) Resultant posterior for a single run of the multi-round SNPE inference on GW strain data. Contours are shown at 68% and 95% confidence intervals, and dashed lines indicate the fiducial parameter values. The fake observation $\mathbf{x}_0$ was generated with the same simulator function as the training forward model.

$\mathcal{M} \equiv (m_1 m_2)^{\frac{3}{5}} / (m_1 + m_2)^{\frac{1}{5}}$ and $q \equiv m_2 / m_1 \leq 1$ – and spin characteristics (Racine 2008).

We established baseline values for our simulated event $\mathbf{x}_0$ at $\{d_L = 390 \text{ Mpc}, q = 0.79, \mathcal{M} = 30.2 M_\odot, \chi_{\text{eff}} = -0.09\}$, akin to the GW150914 event as detailed in the first column of Table 1 by Abbott *et al.* (2016). Following the procedure described in Karathanasis *et al.* (2023), the defined priors are uniform ranging from $[100.0 \text{ Mpc}, 0.2, 12.0 M_\odot, -1.0]$ to $[1000.0 \text{ Mpc}, 1.0, 45.0 M_\odot, 1.0]$, with narrower margins for chirp mass and luminosity distance to facilitate a straightforward example. For density estimation, we utilize the `sbi` backend with a MAF and a 1D convolutional embedding network, undergoing five rounds of training with 1000 simulations per round.

Figure 8b illustrates the progression of the cumulative posterior probability, as defined in Equation 4, as a function of training time. The noticeable increase in probability density for both training and validation datasets is attributed to the model’s focus on learning the posterior density near the target observation $\mathcal{P}(\theta|\mathbf{x} = \mathbf{x}_0)$. However, an early increase in validation loss during subsequent rounds suggests the potential for overfitting, indicating the model’s limitations in gleaning further insights from the simulation inputs. Sampling the posterior at $\mathbf{x} = \mathbf{x}_0$ depicted in Figure 8c, reveals that we are able to constrain each of the four parameters around the fiducial point. For this particular GW example, using LTU-ILI

allows us to parameterize the simulator (waveform approximant, number of detectors considered, geometry of the detection, duration of the signal³, ...), the architectures to train and the metrics to compute with one single set of parameter files, thus making the organization and replication of unified inference tests easier for this particularly difficult problem.

5.5. Inferring Dust Parameters

LTU-ILI is a practical tool for exploratory discovery, especially useful for investigating novel datasets within which likelihoods are not well-defined. In this application, we study how different observables can be used to constrain different degeneracies within galactic dust models, and how their observational combinations can be used to maximally extract information within an ILI model. Dust makes up approximately 1% of the interstellar medium, but reprocesses around 30% of stellar emission (Silva *et al.* 1998). Understanding the properties of dust and its impact on observables is, therefore, a key goal of galactic and extragalactic astronomy, and a fundamental uncertainty in downstream cosmological inference analyses (Crocce *et al.* 2016). Dust leads to an overall reddening of the emission, parameterized by

³ Note, the frequency resolution is only determined by the time duration of the signal, whereas the sampling rate only gives the Nyquist frequency

the optical depth and attenuation law; these are analogous to the normalization and slope, particularly where the latter is parameterized as a power law. Below we show how these parameters can be inferred from observational properties using a simple dust model (Trayford *et al.* 2015) applied to the Simba cosmological hydrodynamic simulation (Davé *et al.* 2019) within an ILI framework.

We first briefly describe our forward model for the stellar emission and dust attenuation. We generated spectra for all galaxies with stellar mass $> 10^{10} M_{\odot}$ in the Simba simulation using **Synthesizer** (Lovell *et al. in prep.*)⁴. The integrated intrinsic stellar emission of each galaxy was obtained by linking each stellar particle to the BC03 stellar population synthesis models (Bruzual and Charlot 2003) based on their age and metallicity. We then modelled the dust attenuation as a wavelength-dependent power law, with slope α ,

$$T(\lambda, t) = \exp \left[\tau(t) \left(\frac{\lambda}{\lambda_V} \right)^{-\alpha} \right], \quad (15)$$

where λ_V is the wavelength of the V band (550 nm). Here the optical depth τ is dependent on the age of the stellar population t ; star particles younger than 10 Myr are assumed to still reside within their birth clouds (BC), and therefore experience an extra source of attenuation,

$$\tau(t) = \begin{cases} \tau_{BC} + \tau_{ISM} & ; \quad t \leq 10 \text{ Myr} \\ \tau_{ISM} & ; \quad t > 10 \text{ Myr} \end{cases} \quad (16)$$

Assuming the same underlying stellar emission model, we generated rest frame optical photometry in the g and r bands for all galaxies in Simba for 1000 simulations on a Latin hypercube, with flat priors on the parameters in the range $\alpha \in [0.5, 2.0]$, $\tau_{ISM} \in [0.01, 0.5]$ and $\tau_{BC} \in [0.3, 1.5]$. We then take these simulations and measure the r -band luminosity function and the $g - r$ color distribution, and use these as our summary statistics. We perform NPE using an ensemble of a MAF with 50 hidden features and five neural transformations and an MDN with 50 hidden features and five mixture components.

Figure 9 shows the posteriors on a test parameter – data pair. We show results from training on just the luminosity function, just the color, and the two combined. It is clear that the overall normalisation of the luminosity function constrains the ISM optical depth, τ_{ISM} , whereas the colour constrains the slope of the attenuation law, α . When combined, these parameters are simultaneously tightly constrained. The birth cloud attenuation has a more subtle effect on the luminosity function and colour distribution, but the combination leads to tighter constraints. LtU-ILI greatly simplifies the setup of the inference pipelines used here, for rapid testing and comparison of combinations of data sources.

5.6. The Mass and Energy Loading of Galactic Winds

Lastly, we demonstrate how using custom embedding networks within LtU-ILI can aid posterior recovery for low signal-to-noise problems such as inferring parameters of semi-analytic models of galaxy formation. One such

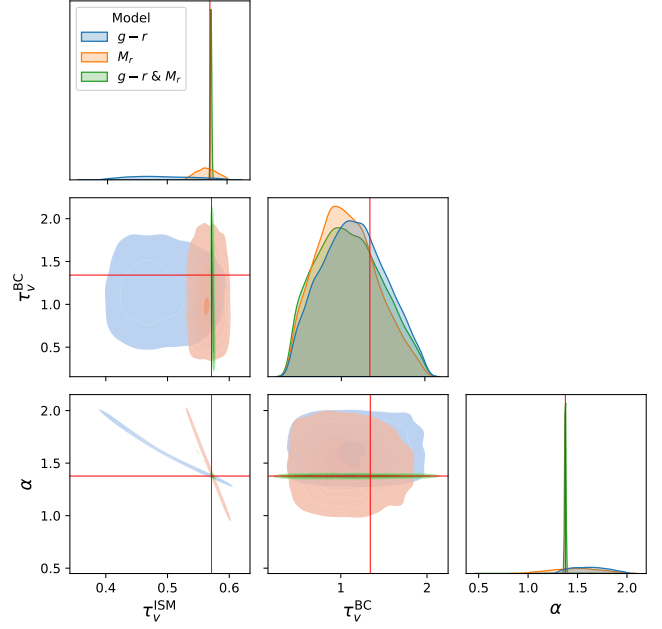


FIG. 9.— Dust parameter inference example on the Simba simulation. Example posteriors on the ISM and birth cloud optical depth as well as the slope of the attenuation law, from the distribution of $g - r$ colors (blue), the r -band magnitude luminosity function (orange), and the combination of both (green).

example is studying the poorly constrained role of feedback in regulating star formation and shaping the stellar-to-halo-mass relation. Recently, Carr *et al.* (2023) and Pandya *et al.* (2023) showed that supernova-driven galactic winds can heat and stir turbulence in the circumgalactic medium of dwarf galaxies and Milky Way-mass halos. This can suppress the cooling and accretion of gas onto galaxies thereby helping to regulate star formation. The strength of this “preventative feedback” is controlled by two wind parameters: the mass loading factor (η_M) and the energy loading factor (η_E). Together, these describe the specific energy of winds and summarize the fraction of mass and energy produced by supernovae that leave the galaxy and become available for large-scale heating.

Here we want to explore the connection between η_M , η_E , and the stellar-to-halo-mass relation predicted by numerous realizations of the next-generation semi-analytic model **sapphire** (Pandya *et al. in prep.*). The model used here is more similar to Carr *et al.* (2023) than Pandya *et al.* (2023) in that it neglects turbulence and assumes purely thermal heating of the circumgalactic medium (CGM) for simplicity. We assume that η_M and η_E both follow power law scalings with halo virial velocity V_{vir} (a proxy for halo mass):

$$\eta_M = A_M \left(\frac{V_{\text{vir}}}{125 \text{ km s}^{-1}} \right)^{\alpha_M} \quad (17)$$

$$\eta_E = A_E \left(\frac{V_{\text{vir}}}{125 \text{ km s}^{-1}} \right)^{\alpha_E} \quad (18)$$

Here, A_M , α_M , A_E and α_E are hyperparameters that govern these power law scalings, and we have ignored any redshift dependence. We generate 5,000 model realizations on a Latin hypercube by uniformly sampling combinations of $A_M \in [0.01, 100]$, $\alpha_M \in [-2, 2]$, $A_E \in [0.01, 1]$

⁴ <https://flaresimulations.github.io/synthesizer>

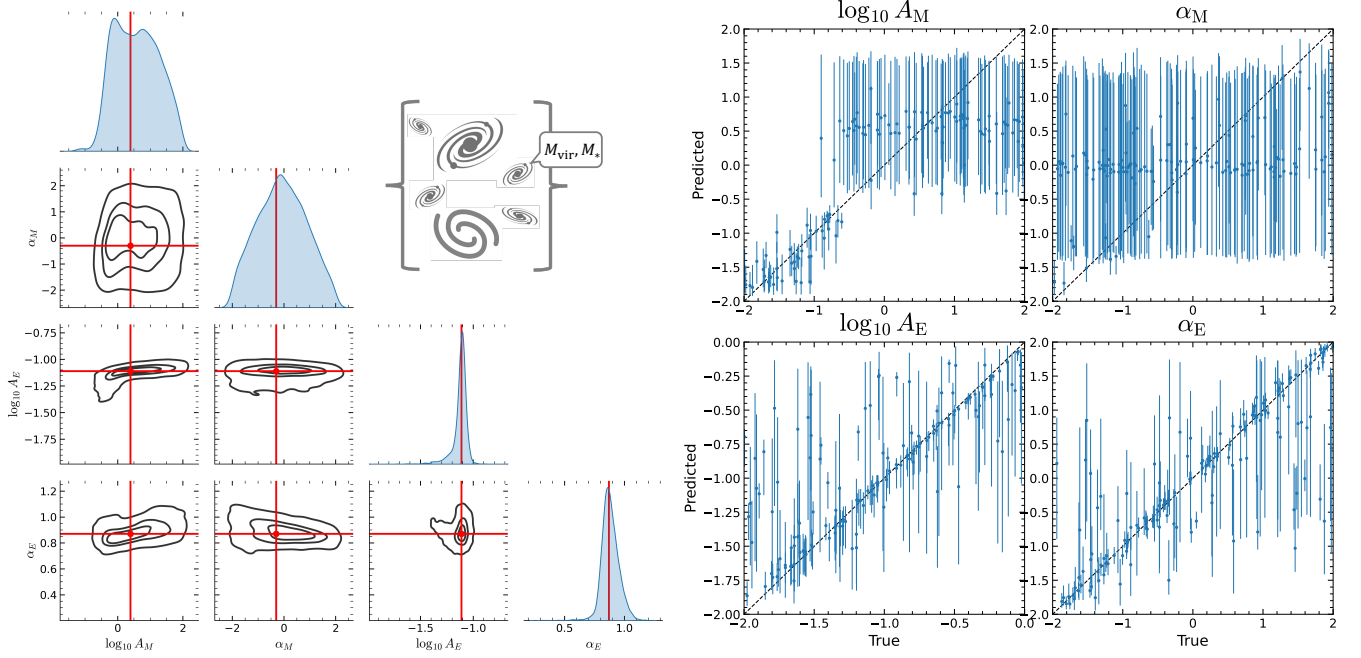


FIG. 10.— Inference of mass and energy loading parameters of **sapphire** semi-analytic model from sets of galaxy virial and stellar masses, *i.e.*, $\mathbf{x} = \{(M_{\text{vir}}, M_*, \frac{M_{\text{vir}}}{M_*})\}_{i=1}^{n_x}$. The left plots show contours of an example inferred posterior, while the right plots show the true values vs. predictions across the full test set. Here, we utilize the **fishnets** architecture to aggregate sets of galaxy properties before performing NPE. Posterior predictive plots show our NPE network is sensitive to both energy loading hyper-parameters but cannot learn the mass loading hyper-parameters with the $M_* - M_{\text{vir}}$ relation alone.

and $\alpha_E \in [-2, 2]$. All other parameters of the model were fixed to reasonable values following Pandya *et al.* (2023). With these parameters, the system of ordinary differential equations describing the model (Equations 1, 2, and 4-8 of Pandya *et al.* 2023) is evolved using mass accretion histories of 120 halos randomly selected from five subvolumes of the TNG100 simulation (Pillepich *et al.* 2018). We find that this is a sufficient number of halos to capture variations in the stellar-to-halo-mass relation with model parameters. We restrict to halos with $\log M_{\text{vir}}/M_\odot = 10 - 12.4$ at $z = 0$, spanning the dwarf and Milky Way regime in which supernova feedback is thought to dominate.

As an embedding network, we use the **fishnets** architecture by Makinen *et al.* (2023) to extract what information the stellar-to-halo-mass relation contains about $\theta = (A_M, \alpha_M, A_E, \alpha_E)$. The output from each **sapphire** realization is a set of $n_x = 120$ independently-evolved galaxies, *i.e.*, $\mathbf{x} = \{(M_{\text{vir}}, M_*, \frac{M_{\text{vir}}}{M_*})\}_{i=1}^{n_x}$ where M_{vir} and M_* are the $z = 0$ halo mass and stellar mass, respectively. The likelihood here is a product of each individual galaxies' likelihoods, $\mathcal{L}(\{x_i\}|\theta) = \prod_{i=1}^{n_x} \mathcal{L}(x_i|\theta)$, requiring an aggregation over heterogeneous data distributions. Following the optimal **fishnets** formalism presented in Makinen *et al.* (2023), we embed data into neural score embeddings and Fisher weights, each parameterized by fully-connected networks of size [128, 128, 128] with **LeakyRelu** activations before passing the weighted scores to two NPE networks, a Neural Spline Flow (Durkan *et al.* 2019) and a Gaussianization Flow (Meng *et al.* 2020).

We show the resulting true parameter versus predicted posterior plots in Figure 10. We find that galaxy energy loading factors are sensitive to halo and stellar mass

properties as expected, while mass loading factors prove more difficult to constrain. This is consistent with Carr *et al.* (2023) who showed that the stellar-to-halo-mass relation is largely insensitive to variations in η_M but very responsive to changes in η_E . The physical reason for this is that increasing η_M leads to a higher CGM density and enhanced cooling of gas back into the galaxy without suppressing star formation, thus leading to a similar $M_* - M_{\text{vir}}$ relation. Instead, increasing η_E can heat the CGM, prevent gas accretion, reduce star formation and alter the normalization and shape of the $M_* - M_{\text{vir}}$ relation. Interestingly, for **sapphire** realizations run at $A_M \lesssim 0.25$ there is some information available to constrain mass-loading parameters, which isn't accessible for high A_M . LTU-ILI is able to capture this effect and carefully estimate posterior error bars, demonstrating its ability to calibrate inference both in low and high signal-to-noise regimes. In the future, it will be compelling to use **fishnets** to understand what additional galaxy properties output by **sapphire** contain the information needed to better learn both η_M and η_E .

6. DISCUSSION

ILI methods are guaranteed to recover the true posterior under the assumptions that:

1. There is a sufficient amount of training data to cover data and parameter space.
2. The training data or simulator is reflective of reality.
3. The chosen neural architecture is sufficiently flexible to capture the data-parameter relationship.

Despite the best practices implemented in LTU-ILI, biases may occur when these assumptions are violated. Be-

fore applying these approaches to real observational data, it is crucial to understand these failure modes, as well as how to build inference pipelines to be robust to them. Below, we review important considerations provide qualitative guidance for using ILI in observational settings.

Firstly, model misspecification occurs when the data-generating function for creating training data is poorly representative of physical processes in the real world (Cannon *et al.* 2022). This can cause an implicit model to interpret the wrong data-parameter connection, resulting in biased predictions. In astrophysics and cosmology, combating model misspecification often requires running more realistic, high-resolution simulators which can be more computationally demanding. In turn, this reduces the number of available simulations, damaging assumption (1). Alternative approaches include increasing the noise model of a given simulation (*e.g.*, Modi *et al.* 2023), training NDEs with a conservative loss to inflate uncertainties and counteract misspecification (*e.g.*, Kelly *et al.* 2023; Huang *et al.* 2023), or using latent functions present within intermediate layers of the Bayesian hierarchical model as diagnostics (*e.g.*, Leclercq 2022).

A more direct approach to addressing model misspecification is through Bayesian model comparison (Silva *et al.* 1998; Handley *et al.* 2015). Given observational data $\mathbf{x}_o$, this technique seeks to quantify whether a proposed data-generating model $\mathcal{M}(\theta)$ is reflective of reality. This is often done through calculation of the model evidence, $p(\mathbf{x}_o|\mathcal{M})$, which, given an explicit likelihood model $\mathcal{L}(\mathbf{x}|\theta, \mathcal{M})$, can be done numerically, though requiring massive simulation budgets. Recent ILI approaches using the harmonic mean estimator (McEwen *et al.* 2021; Spurio Mancini *et al.* 2023) or Evidence Networks (Jeffrey and Wandelt 2024) have been shown to greatly accelerate these calculations. They also allow us to learn distributions implicitly from simulations, eliminating the need for explicit likelihoods. These novel evidence estimators are a natural addition to LTU-ILI and are planned for a future release. When implemented, these techniques will allow users to form quantitative arguments to addressing model misspecification.

Secondly, even for well-specified models, ILI methods may overfit and suffer from poor out-of-sample performance when training data is limited. This behavior has led to recent discussion as to whether ILI methods are systematically overconfident (Hermans *et al.* 2022). PIT tests (Section 2.5) on independent validation sets are a useful diagnostic tool for detecting overconfidence. For a fixed simulation budget, this problem can be mitigated through the use of ensembling (Hermans *et al.* 2022). Ensembling averages the predictions of multiple, independently trained models together, naturally inflating predictive uncertainty to account for epistemic bias (Lakshminarayanan *et al.* 2017). Alternatively, approximate Bayesian neural networks (Blundell *et al.* 2015; Gal and Ghahramani 2016) seek to constrain posteriors over model parameters during training, which they then subsequently propagate to model outputs to increase posterior variance. Lastly, once a model has been trained, its predictions can be reweighted through frequentist recalibration (Masserano *et al.* 2022) which guarantees accurate posterior coverage over held-out test data. While approximate Bayesian networks and frequentist recalibration are not yet implemented in LTU-ILI, their inte-

gration is considered for future versions of the codebase.

Overfitting can also be calibrated through careful hyperparameter selection. Choices of input data, neural architectures, and training procedures all play a quantifiable role in the final accuracy and out-of-sample bias measured in an ILI model. Oftentimes, selection of hyperparameters is a trade-off between choosing a flexible, expressive model which produces high-accuracy but is prone to overfitting, against a shallow, weaker model with more generalizable predictions. This is commonly referred to as the ‘bias-variance tradeoff’ (Belkin *et al.* 2019), and can be interpreted as placing assumptions (1) and (3) in direct tension. To address this, practitioners often perform automated searches over hyperparameters to find configurations which minimize variance within a given bias tolerance. Often, these searches are accelerated through the use of Bayesian optimization tools such as `optuna` (Akiba *et al.* 2019). These can be used to motivate ILI design choices directly from data, a necessary step for scientific studies. In the LTU-ILI repository, we provide examples of hyperparameter optimization through grid searching, and we plan to link LTU-ILI to optimal search tools such as `optuna` in future iterations.

Lastly, implicit inference can be very challenging in the high-dimensional parameter regime (*i.e.*, when $\dim(\theta) \gg 100$). As the dimensionality of θ increases, the number of parameters necessary to specify a highly-correlated posterior increases exponentially. This problem is of particular relevance in astrophysics and cosmology, in which we often would like to infer voxelized 2D or 3D fields (*e.g.*, Jasche and Wandelt 2013; de Andres *et al.* 2024). Recent empirical evidence suggests that Moment Networks (Jeffrey and Wandelt 2020) and diffusion-based generative models (Song *et al.* 2020; Legin *et al.* 2024) are remarkably effective neural architectures for these tasks. Though LTU-ILI does not currently include these models, we intend to implement them in future work.

7. CONCLUSION

In conclusion, we have introduced LTU-ILI, a comprehensive framework designed for implicit inference in scientific applications. LTU-ILI provides extensive routines to facilitate the training of neural networks for regression posterior inference, aimed in particular at cases in which the likelihood is intractable. The code encapsulates best practices and state-of-the-art models, ensuring adaptability to a broad spectrum of scientific domains. By consolidating the training, testing, and robustness assessment of ILI models into a unified, modular package, LTU-ILI streamlines the process of addressing novel inference problems while providing for production-level inference and hyperparameter searches, making it accessible to researchers both experienced and novice.

This manuscript is designed to serve as a reference document for future applications of LTU-ILI. In summary, we record detailed theoretical descriptions of every flavor of ILI (Section 2), describe the important functionalities of the codebase (Section 3), and provide guidance for experimental design amidst a broad landscape of estimators (Section 2.3) and backends (Section 3.4). We also have provided benchmarks and science examples on a variety of astronomy and cosmology inference problems. Our results demonstrate that LTU-ILI yields competitive con-

straints on both synthetic and real science examples, indicating that these methods are ready out-of-the-box for application to new problems.

Looking ahead, we aim to apply this framework to modern analyses of the problems described in Section 5, with an emphasis on observational applications to survey data from surveys such as SDSS, Planck, VRO, Euclid, DESI, and JWST. We aim to provide exhaustive benchmarks on these problems, contributing to the formalization of machine learning analyses in astronomy and beyond. We also intend to integrate LTU-ILI with state-of-the-art hyperparameter tuning (*e.g.*, Akiba *et al.* 2019) to provide an automatic, yet principled way for doing model selection, as in Choustikov *et al.* (2024). This includes developing and implementing consolidated metrics for calibrating posterior coverage (*e.g.*, Zhao *et al.* 2021; Lemos *et al.* 2024), to guarantee accurate posterior coverage through hyperparameter search. An alternative approach is to implement recalibration methods (*e.g.*, Masserano *et al.* 2022) which correctly calibrate posteriors given an independent validation set.

We note that, while the current release of LTU-ILI is extensive, we hope to further improve the toolkit by incorporating additional Bayesian machine learning methods. For example, we do not currently include newer ILI procedures such as Truncated Marginal Neural Ratio

Estimation (TMNRE; Miller *et al.* 2021, 2022) or score-based diffusion networks (*e.g.*, Sharrock *et al.* 2022; Dax *et al.* 2023; Linhart *et al.* 2024), each of which has been shown to greatly improve upon the benchmarks that were investigated in Section 4. Also, during the development of this work, **nbi** (Zhang *et al.* 2023) was released as a neural posterior estimation toolkit focused on spectroscopy and light-curve analysis. Though we do not directly integrate **nbi**, we remark that the functionality of **nbi** is already encompassed and extended by LTU-ILI. Lastly, a major future addition to LTU-ILI will involve integration with implicit evidence estimators such as Evidence Networks (Jeffrey and Wandelt 2024) and **harmonic** (McEwen *et al.* 2021; Spurio Mancini *et al.* 2023). With these features, users will be able to do Bayesian posterior inference and model comparison, all within one interface. We reserve implementation and analysis of each of these improvements to future work.

We thank Rosa Malandrino, Niall Jeffrey, and Justin Alsing for useful comments and suggestions on developing the codebase and writing the manuscript. This project was developed as part of the Simons Collaboration on “Learning the Universe.” The Flatiron Institute is supported by the Simons Foundation.

REFERENCES

- E. D. Feigelson and G. J. Babu, *Modern statistical methods for astronomy: with R applications* (Cambridge University Press, 2012).
- S. Dodelson and F. Schmidt, *Modern cosmology* (Academic press, 2020).
- G. M. Eadie, J. S. Speagle, J. Cisewski-Kehe, D. Foreman-Mackey, D. Huppenkothen, D. E. Jones, A. Springford, and H. Tak, arXiv preprint arXiv:2302.04703 (2023).
- National Academies of Sciences, Engineering, and Medicine, *Pathways to Discovery in Astronomy and Astrophysics for the 2020s* (The National Academies Press, Washington, DC, 2023).
- G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, *Reviews of Modern Physics* **91**, 045002 (2019).
- D. Huppenkothen, M. Ntampaka, M. Ho, M. Fouesneau, B. Nord, J. Peek, M. Walmsley, J. F. Wu, C. Avestruz, T. Buck, *et al.*, arXiv preprint arXiv:2310.12528 (2023).
- K. Cranmer, J. Brehmer, and G. Louppe, *Proceedings of the National Academy of Sciences* **117**, 30055 (2020).
- J.-M. Marin, P. Pudlo, C. P. Robert, and R. J. Ryder, *Statistics and computing* **22**, 1167 (2012).
- N. Christensen, R. Meyer, L. Knox, and B. Luey, *Classical and Quantum Gravity* **18**, 2677 (2001).
- S. Brooks, A. Gelman, G. Jones, and X.-L. Meng, *Handbook of markov chain monte carlo* (CRC press, 2011).
- N. Jeffrey, J. Alsing, and F. Lanusse, *Monthly Notices of the Royal Astronomical Society* **501**, 954 (2021).
- T. L. Mäkinen, T. Charnock, J. Alsing, and B. D. Wandelt, *Journal of Cosmology and Astroparticle Physics* **2021**, 049 (2021).
- T. L. Mäkinen, T. Charnock, P. Lemos, N. Porqueres, A. F. Heavens, and B. D. Wandelt, *The Open Journal of Astrophysics* **5** (2022a), 10.21105/astro.2207.05202.
- N. S. de Santi, H. Shao, F. Villaescusa-Navarro, L. R. Abramo, R. Teyssier, P. Villanueva-Domingo, Y. Ni, D. Anglés-Alcázar, S. Genel, E. Hernandez-Martinez, *et al.*, arXiv preprint arXiv:2302.14101 (2023a).
- C. Hahn, P. Lemos, L. Parker, B. R.-S. Blancard, M. Eickenberg, S. Ho, J. Hou, E. Massara, C. Modi, A. M. Dizgah, *et al.*, arXiv preprint arXiv:2310.15246 (2023a).
- M. Dax, S. R. Green, J. Gair, J. H. Macke, A. Buonanno, and B. Schölkopf, *Phys. Rev. Lett.* **127**, 241103 (2021a), arXiv:2106.12594 [gr-qc].
- D. H. Cheung, K. W. Wong, O. A. Hannuksela, T. G. Li, and S. Ho, *Physical Review D* **106**, 083014 (2022).
- M. Ho, M. Ntampaka, M. M. Rau, M. Chen, A. Lansberry, F. Ruehle, and H. Trac, *Nature Astronomy* **6**, 936 (2022).
- D. de Andres, W. Cui, F. Ruppin, M. De Petris, G. Yepes, G. Gianfagna, I. Lahouli, G. Aversano, R. Dupuis, M. Jarraya, *et al.*, *Nature Astronomy* **6**, 1325 (2022).
- M. Walmsley, L. Smith, C. Lintott, Y. Gal, S. Bamford, H. Dickinson, L. Fortson, S. Kruk, K. Masters, C. Scarlata, *et al.*, *Monthly Notices of the Royal Astronomical Society* **491**, 1554 (2020).
- A. Ghosh, C. M. Urry, A. Rau, L. Perreault-Levasseur, M. Cranmer, K. Schawinski, D. Stark, C. Tian, R. Ofman, T. T. Ananna, *et al.*, *The Astrophysical Journal* **935**, 138 (2022).
- J. Hermans, N. Banik, C. Weniger, G. Bertone, and G. Louppe, *Monthly Notices of the Royal Astronomical Society* **507**, 1999 (2021).
- J. Alvey, M. Gerdes, and C. Weniger, arXiv preprint arXiv:2304.02032 (2023).
- J. G. Rogers, C. Janó Muñoz, J. E. Owen, and T. L. Mäkinen, *Monthly Notices of the Royal Astronomical Society* **519**, 6028 (2023).
- M. Aubin, C. Cuesta-Lazaro, E. Tregidga, J. Viaña, C. Garraffo, I. E. Gordon, M. López-Morales, R. J. Hargreaves, V. Y. Makhnev, J. J. Drake, *et al.*, arXiv preprint arXiv:2309.09337 (2023).
- Y. Gal and Z. Ghahramani, in *international conference on machine learning* (PMLR, 2016) pp. 1050–1059.
- B. Lakshminarayanan, A. Pritzel, and C. Blundell, *Advances in neural information processing systems* **30** (2017).
- W. J. Maddox, P. Izmailov, T. Garipov, D. P. Vetrov, and A. G. Wilson, *Advances in neural information processing systems* **32** (2019).
- P. Lemos, M. Cranmer, M. Abidi, C. Hahn, M. Eickenberg, E. Massara, D. Yallup, and S. Ho, *Machine Learning: Science and Technology* **4**, 01LT01 (2023), arXiv:2207.08435 [astro-ph.CO].

- H. Jin, Q. Song, and X. Hu, in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (2019) pp. 1946–1956.
- C. White, W. Neiswanger, and Y. Savani, in *Proceedings of the AAAI Conference on Artificial Intelligence* (2021).
- B. Wang, J. Leja, V. A. Villar, and J. S. Speagle, *ApJ* **952**, L10 (2023), arXiv:2304.05281 [astro-ph.IM].
- C. Modi, S. Pandey, M. Ho, C. Hahn, B. Blancard, and B. Wandelt, arXiv preprint arXiv:2309.15071 (2023).
- A. Tejero-Cantero, J. Boelts, M. Deistler, J.-M. Lueckmann, C. Durkan, P. J. Gonçalves, D. S. Greenberg, and J. H. Macke, *Journal of Open Source Software* **5**, 2505 (2020).
- J. Alsing, T. Charnock, S. Feeney, and B. Wandelt, *Monthly Notices of the Royal Astronomical Society* **488**, 4440 (2019).
- F. Rozet, A. Delaunoy, B. Miller, *et al.*, “LAMPE: Likelihood-free amortized posterior estimation,” (2021).
- P. Lemos, A. Coogan, Y. Hezaveh, and L. P. Levasseur, in *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, Proceedings of Machine Learning Research, Vol. 202, edited by A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett (PMLR, 2023) pp. 19256–19273.
- M. Tegmark, M. R. Blanton, M. A. Strauss, F. Hoyle, D. Schlegel, R. Scoccimarro, M. S. Vogeley, D. H. Weinberg, I. Zehavi, A. Berlind, T. Budavari, A. Connolly, D. J. Eisenstein, D. Finkbeiner, J. A. Frieman, J. E. Gunn, A. J. S. Hamilton, L. Hui, B. Jain, D. Johnston, S. Kent, H. Lin, R. Nakajima, R. C. Nichol, J. P. Ostriker, A. Pope, R. Scranton, U. Seljak, R. K. Sheth, A. Stebbins, A. S. Szalay, I. Szapudi, L. Verde, Y. Xu, J. Annis, N. A. Bahcall, J. Brinkmann, S. Burles, F. J. Castander, I. Csabai, J. Loveday, M. Doi, M. Fukugita, I. Gott, J. Richard, G. Hennessy, D. W. Hogg, Ž. Ivezić, G. R. Knapp, D. Q. Lamb, B. C. Lee, R. H. Lupton, T. A. McKay, P. Kunszt, J. A. Munn, L. O’Connell, J. Peoples, J. R. Pier, M. Richmond, C. Rockosi, D. P. Schneider, C. Stoughton, D. L. Tucker, D. E. Vanden Berk, B. Yanny, D. G. York, and SDSS Collaboration, *ApJ* **606**, 702 (2004), arXiv:astro-ph/0310725 [astro-ph].
- N. Christensen and R. Meyer, *Reviews of Modern Physics* **94**, 025001 (2022).
- N. Porqueres, A. Heavens, D. Mortlock, and G. Lavaux, *Monthly Notices of the Royal Astronomical Society* **509**, 3194 (2022).
- F. Boese and S. Doebereiner, *Astronomy & Astrophysics* **370**, 649 (2001).
- J. Braun, J. Dumm, F. De Palma, C. Finley, A. Karle, and T. Montaruli, *Astroparticle Physics* **29**, 299 (2008).
- E. Tsaprazi, J. Jasche, G. Lavaux, and F. Leclercq, arXiv preprint arXiv:2301.03581 (2023).
- L. Dingeldein, P. Cossio, and R. Covino, *Machine Learning: Science and Technology* **4**, 025009 (2023).
- M. Ntampaka, J. ZuHone, D. Eisenstein, D. Nagai, A. Vikhlinin, L. Hernquist, F. Marinacci, D. Nelson, R. Pakmor, A. Pillepich, *et al.*, *The Astrophysical Journal* **876**, 82 (2019).
- J. Fluri, T. Kacprzak, A. Lucchi, A. Schneider, A. Refregier, and T. Hofmann, *Phys. Rev. D* **105**, 083518 (2022).
- F. Lanasue, Q. Ma, N. Li, T. E. Collett, C.-L. Li, S. Ravanbakhsh, R. Mandelbaum, and B. Póczos, *Monthly Notices of the Royal Astronomical Society* **473**, 3895 (2018).
- D. B. Rubin, *The Annals of Statistics* **12**, 1151 (1984).
- C. M. Schafer and P. E. Freeman, in *Statistical Challenges in Modern Astronomy V* (Springer, 2012) pp. 3–19.
- E. Cameron and A. Pettitt, *Monthly Notices of the Royal Astronomical Society* **425**, 44 (2012).
- C. Hahn, M. Vakili, K. Walsh, A. P. Hearin, D. W. Hogg, and D. Campbell, *Monthly Notices of the Royal Astronomical Society* **469**, 2791 (2017).
- J. Alsing, B. Wandelt, and S. Feeney, *MNRAS* **477**, 2874 (2018), arXiv:1801.01497 [astro-ph.CO].
- G. Papamakarios, arXiv preprint arXiv:1910.13233 (2019).
- C. M. Bishop, *Mixture density networks*, WorkingPaper (Aston University, 1994).
- G. Papamakarios and I. Murray, *Advances in neural information processing systems* **29** (2016).
- G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, *The Journal of Machine Learning Research* **22**, 2617 (2021).
- D. Greenberg, M. Nonnenmacher, and J. Macke, in *International Conference on Machine Learning* (PMLR, 2019) pp. 2404–2414.
- M. Vasisht, F. Rozet, O. Absil, P. Mollière, E. Nasedkin, and G. Louppe, *A&A* **672**, A147 (2023), arXiv:2301.06575 [astro-ph.EP].
- M. Crisostomi, K. Dey, E. Barausse, and R. Trotta, *Phys. Rev. D* **108**, 044029 (2023), arXiv:2305.18528 [gr-qc].
- J. Alsing, T. Charnock, S. Feeney, and B. Wandelt, *MNRAS* **488**, 4440 (2019), arXiv:1903.00007 [astro-ph.CO].
- G. Papamakarios, D. Sterratt, and I. Murray, in *The 22nd International Conference on Artificial Intelligence and Statistics* (PMLR, 2019) pp. 837–848.
- C. P. Robert, G. Casella, and G. Casella, *Monte Carlo statistical methods*, Vol. 2 (Springer, 1999).
- D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, *Journal of the American statistical Association* **112**, 859 (2017).
- C. P. Robert, G. Casella, C. P. Robert, and G. Casella, *Monte Carlo statistical methods*, 267 (2004).
- N. Jeffrey, J. Alsing, and F. Lanasue, *MNRAS* **501**, 954 (2021), arXiv:2009.08459 [astro-ph.CO].
- J. Brehmer, F. Kling, I. Espejo, and K. Cranmer, *Comput. Softw. Big Sci.* **4**, 3 (2020), arXiv:1907.10621 [hep-ph].
- J. Hermans, V. Begy, and G. Louppe, in *International conference on machine learning* (PMLR, 2020) pp. 4239–4248.
- K. Cranmer, J. Pavez, and G. Louppe, arXiv preprint arXiv:1506.02169 (2015).
- A. Cole, B. K. Miller, S. J. Witte, M. X. Cai, M. W. Grootes, F. Nattino, and C. Weniger, *J. Cosmology Astropart. Phys.* **2022**, 004 (2022), arXiv:2111.08030 [astro-ph.CO].
- K. Karchev, R. Trotta, and C. Weniger, *MNRAS* **520**, 1056 (2023), arXiv:2209.06733 [astro-ph.CO].
- B. Miller, A. Cole, P. Forré, G. Louppe, and C. Weniger, *Advances in Neural Information Processing Systems* **34**, 129 (2021), arXiv:2107.01214 [stat.ML].
- A. Delaunoy, A. Wehenkel, T. Hinderer, S. Nissanke, C. Weniger, A. R. Williamson, and G. Louppe, arXiv e-prints, arXiv:2010.12931 (2020), arXiv:2010.12931 [astro-ph.IM].
- U. Bhardwaj, J. Alvey, B. K. Miller, S. Nissanke, and C. Weniger, *Phys. Rev. D* **108**, 042004 (2023), arXiv:2304.02035 [gr-qc].
- B. K. Miller, A. Cole, C. Weniger, F. Nattino, O. Ku, and M. W. Grootes, *J. Open Source Softw.* **7**, 4205 (2022).
- Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, arXiv preprint arXiv:2011.13456 (2020).
- J.-M. Lueckmann, J. Boelts, D. Greenberg, P. Gonçalves, and J. Macke, in *International conference on artificial intelligence and statistics* (PMLR, 2021) pp. 343–351.
- D. Watson-Parris, A. Williams, L. Deaconu, and P. Stier, *Geoscientific Model Development* **14**, 7659 (2021).
- M. U. Gutmann, J. Cor, *et al.*, *Journal of Machine Learning Research* **17**, 1 (2016).
- F. Leclercq, *Phys. Rev. D* **98**, 063511 (2018), arXiv:1805.07152 [astro-ph.CO].
- D. Lopez-Paz and M. Oquab, arXiv preprint arXiv:1610.06545 (2016).
- S. R. Cook, A. Gelman, and D. B. Rubin, *Journal of Computational and Graphical Statistics* **15**, 675 (2006).
- D. Zhao, N. Dalmaso, R. Izbicki, and A. B. Lee, in *Uncertainty in Artificial Intelligence* (PMLR, 2021) pp. 1830–1840.
- R. Bordoloi, S. J. Lilly, and A. Amara, *Monthly Notices of the Royal Astronomical Society* **406**, 881 (2010).
- M. Tanaka, J. Coupon, B.-C. Hsieh, S. Mineo, A. J. Nishizawa, J. Speagle, H. Furusawa, S. Miyazaki, and H. Murayama, *Publications of the Astronomical Society of Japan* **70**, S9 (2018).
- S. Talts, M. Betancourt, D. Simpson, A. Vehtari, and A. Gelman, arXiv e-prints, arXiv:1804.06788 (2018), arXiv:1804.06788 [stat.ME].
- A. Y. Wang, A. Mittal, C. Brooks, and S. Oney, *Proceedings of the ACM on Human-Computer Interaction* **3**, 1 (2019).
- D. Singh and B. Singh, *Applied Soft Computing* **97**, 105524 (2020).
- D. P. Kingma and J. Ba, arXiv preprint arXiv:1412.6980 (2014).
- Y. Bai, E. Yang, B. Han, Y. Yang, J. Li, Y. Mao, G. Niu, and T. Liu, *Advances in Neural Information Processing Systems* **34**, 24392 (2021).

- M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, in *12th USENIX symposium on operating systems design and implementation (OSDI 16)* (2016) pp. 265–283.
- A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, *Advances in neural information processing systems* **32** (2019).
- T. Charnock, G. Lavaux, and B. D. Wandelt, *Physical Review D* **97**, 083004 (2018).
- J. Alsing and B. Wandelt, *Monthly Notices of the Royal Astronomical Society: Letters* **476**, L60 (2018).
- J. Alsing, B. Wandelt, and S. Feeney, *Monthly Notices of the Royal Astronomical Society* **477**, 2874 (2018).
- J.-M. Lueckmann, P. J. Goncalves, G. Bassetto, K. Öcal, M. Nonnenmacher, and J. H. Macke, *Advances in neural information processing systems* **30** (2017).
- C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios, “nflows: normalizing flows in PyTorch,” (2020).
- F. Rozet *et al.*, *Zuko: Normalizing flows in PyTorch* (2022).
- J. Hermans, A. Delaunoy, F. Rozet, A. Wehenkel, V. Begy, and G. Louppe, *stat.* **1050** (2022).
- C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, in *International conference on machine learning* (PMLR, 2015) pp. 1613–1622.
- A. D. Cobb and B. Jalaian, in *Uncertainty in Artificial Intelligence* (PMLR, 2021) pp. 675–685.
- F. Chollet *et al.*, “Keras,” <https://github.com/fchollet/keras> (2015).
- Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, *Proceedings of the IEEE* **86**, 2278 (1998).
- T. N. Kipf and M. Welling, *arXiv preprint arXiv:1609.02907* (2016).
- D. Foreman-Mackey, D. W. Hogg, D. Lang, and J. Goodman, *Publications of the Astronomical Society of the Pacific* **125**, 306 (2013).
- J. Goodman and J. Weare, *Communications in applied mathematics and computational science* **5**, 65 (2010).
- E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. A. Szerlip, P. Horsfall, and N. D. Goodman, *J. Mach. Learn. Res.* **20**, 28:1 (2019).
- R. M. Neal, *The annals of statistics* **31**, 705 (2003).
- R. M. Neal *et al.*, *Handbook of markov chain monte carlo* **2**, 2 (2011).
- M. D. Hoffman, A. Gelman, *et al.*, *J. Mach. Learn. Res.* **15**, 1593 (2014).
- A. Graves, *Advances in neural information processing systems* **24** (2011).
- D. J. Bartlett, L. Kammerer, G. Kronberger, H. Desmond, P. G. Ferreira, B. D. Wandelt, B. Burlacu, D. Alonso, and M. Zennaro, *arXiv preprint arXiv:2311.15865* (2023).
- D. S. Greenberg, M. Nonnenmacher, and J. H. Macke, *arXiv e-prints*, *arXiv:1905.07488* (2019), *arXiv:1905.07488 [cs.LG]*.
- P. Ramesh, J.-M. Lueckmann, J. Boelts, Á. Tejero-Cantero, D. S. Greenberg, P. J. Gonçalves, and J. H. Macke, *arXiv preprint arXiv:2203.06481* (2022).
- M. Ho, J. Soltis, A. Farahi, D. Nagai, A. Evrard, and M. Ntampaka, *arXiv preprint arXiv:2303.00005* (2023).
- K. Dolag, E. Komatsu, and R. Sunyaev, *Monthly Notices of the Royal Astronomical Society* **463**, 1797 (2016).
- J. Soltis, M. Ntampaka, J. F. Wu, J. Zuhone, A. Evrard, A. Farahi, M. Ho, and D. Nagai, *The Astrophysical Journal* **940**, 60 (2022).
- G. Papamakarios, T. Pavlakou, and I. Murray, *Advances in neural information processing systems* **30** (2017).
- A. Cooray and R. Sheth, *Physics reports* **372**, 1 (2002).
- A. Spurio Mancini, D. Piras, J. Alsing, B. Joachimi, and M. P. Hobson, *Monthly Notices of the Royal Astronomical Society* **511**, 1771 (2022).
- C. Hahn, M. Eickenberg, S. Ho, J. Hou, P. Lemos, E. Massara, C. Modi, A. M. Dizgah, B. Régalo-Saint Blancard, and M. M. Abidi, *Journal of Cosmology and Astroparticle Physics* **2023**, 010 (2023b).
- F. Villaescusa-Navarro, C. Hahn, E. Massara, A. Banerjee, A. M. Delgado, D. K. Ramanah, T. Charnock, E. Giusarma, Y. Li, E. Allys, A. Brochard, C. Uhlemann, C.-T. Chiang, S. He, A. Pisani, A. Obuljen, Y. Feng, E. Castorina, G. Contardo, C. D. Kreisch, A. Nicola, J. Alsing, R. Scoccimarro, L. Verde, M. Viel, S. Ho, S. Mallat, B. Wandelt, and D. N. Spergel, *ApJS* **250**, 2 (2020), *arXiv:1909.05273 [astro-ph.CO]*.
- T. L. Makinen, T. Charnock, P. Lemos, N. Porqueres, A. Heavens, and B. D. Wandelt, *The Open Journal of Astrophysics* **5** (2022b), 10.21105/astro.2207.05202.
- C. Cuesta-Lazaro and S. Mishra-Sharma, *arXiv preprint arXiv:2311.17141* (2023).
- N. S. M. de Santi, H. Shao, F. Villaescusa-Navarro, L. R. Abramo, R. Teyssier, P. Villanueva-Domingo, Y. Ni, D. Anglés-Alcázar, S. Genel, E. Hernández-Martínez, U. P. Steinwandel, C. C. Lovell, K. Dolag, T. Castro, and M. Vogelsberger, *The Astrophysical Journal* **952**, 69 (2023b).
- J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, *Machine learning meets quantum physics*, 199 (2020).
- C. Cutler and É. E. Flanagan, *Phys. Rev. D* **49**, 2658 (1994), *arXiv:gr-qc/9402014 [gr-qc]*.
- E. Thrane and C. Talbot, *PASA* **36**, e010 (2019), *arXiv:1809.02293 [astro-ph.IM]*.
- S. R. Green, C. Simpson, and J. Gair, *Phys. Rev. D* **102**, 104057 (2020), *arXiv:2002.07656 [astro-ph.IM]*.
- S. R. Green and J. Gair, *arXiv e-prints*, *arXiv:2008.03312* (2020), *arXiv:2008.03312 [astro-ph.IM]*.
- M. Dax, S. R. Green, J. Gair, M. Deistler, B. Schölkopf, and J. H. Macke, *arXiv e-prints*, *arXiv:2111.13139* (2021b), *arXiv:2111.13139 [cs.LG]*.
- J. B. Wildberger, M. Dax, S. Buchholz, S. R. Green, J. Macke, and B. Schölkopf, in *Machine Learning for Astrophysics* (2023) p. 34, *arXiv:2305.17161 [cs.LG]*.
- S. Khan, S. Husa, M. Hannam, F. Ohme, M. Pürrer, X. J. Forsteza, and A. Bohé, *Phys. Rev. D* **93**, 044007 (2016), *arXiv:1508.07253 [gr-qc]*.
- G. Ashton, M. Hübner, P. D. Lasky, C. Talbot, K. Ackley, S. Biscoveanu, Q. Chu, A. Divakarla, P. J. Easter, B. Goncharov, F. Hernandez Vivanco, J. Harms, M. E. Lower, G. D. Meadors, D. Melchor, E. Payne, M. D. Pitkin, J. Powell, N. Sarin, R. J. E. Smith, and E. Thrane, *ApJS* **241**, 27 (2019), *arXiv:1811.02042 [astro-ph.IM]*.
- I. M. Romero-Shaw, C. Talbot, S. Biscoveanu, V. D’Emilio, G. Ashton, C. P. L. Berry, S. Coughlin, S. Galaudage, C. Hoy, M. Hübner, K. S. Phukon, M. Pitkin, M. Rizzo, N. Sarin, R. Smith, S. Stevenson, A. Vajpeyi, M. Arène, K. Athar, S. Banagiri, N. Bose, M. Carney, K. Chatziioannou, J. A. Clark, M. Colleoni, R. Cotesta, B. Edelman, H. Estellés, C. García-Quirós, A. Ghosh, R. Green, C. J. Haster, S. Husa, D. Keitel, A. X. Kim, F. Hernandez-Vivanco, I. Magaña Hernandez, C. Karathanasis, P. D. Lasky, N. De Lillo, M. E. Lower, D. Macleod, M. Mateu-Lucena, A. Miller, M. Millhouse, S. Morisaki, S. H. Oh, S. Ossokine, E. Payne, J. Powell, G. Pratten, M. Pürrer, A. Ramos-Buades, V. Raymond, E. Thrane, J. Veitch, D. Williams, M. J. Williams, and L. Xiao, *MNRAS* **499**, 3295 (2020), *arXiv:2006.00714 [astro-ph.IM]*.
- J. Veitch, V. Raymond, B. Farr, W. Farr, P. Graff, S. Vitale, B. Aylott, K. Blackburn, N. Christensen, M. Coughlin, W. Del Pozzo, F. Feroz, J. Gair, C. J. Haster, V. Kalogera, T. Littenberg, I. Mandel, R. O’Shaughnessy, M. Pitkin, C. Rodriguez, C. Röver, T. Sidery, R. Smith, M. Van Der Sluys, A. Vecchio, W. Vousden, and L. Wade, *Phys. Rev. D* **91**, 042003 (2015), *arXiv:1409.7215 [gr-qc]*.
- S. Vitale, R. Lynch, V. Raymond, R. Sturani, J. Veitch, and P. Graff, *Phys. Rev. D* **95**, 064053 (2017), *arXiv:1611.01122 [gr-qc]*.
- É. Racine, *Phys. Rev. D* **78**, 044021 (2008), *arXiv:0803.1820 [gr-qc]*.
- B. P. Abbott *et al.* (LIGO Scientific, Virgo), *Phys. Rev. Lett.* **116**, 241102 (2016), *arXiv:1602.03840 [gr-qc]*.
- C. Karathanasis, B. Revenu, S. Mukherjee, and F. Stachurski, *Astron. Astrophys.* **677**, A124 (2023), *arXiv:2210.05724 [astro-ph.CO]*.
- L. Silva, G. L. Granato, A. Bressan, and L. Danese, *ApJ* **509**, 103 (1998).

- M. Crocce, J. Carretero, A. H. Bauer, A. Ross, I. Sevilla-Noarbe, T. Giannantonio, F. Sobreira, J. Sanchez, E. Gaztanaga, M. C. Kind, *et al.*, *Monthly Notices of the Royal Astronomical Society* **455**, 4301 (2016).
- J. W. Trayford, T. Theuns, R. G. Bower, J. Schaye, M. Furlong, M. Schaller, C. S. Frenk, R. A. Crain, C. D. Vecchia, and I. G. McCarthy, *Monthly Notices of the Royal Astronomical Society* **452**, 2879 (2015), <https://academic.oup.com/mnras/article-pdf/452/3/2879/4922335/stv1461.pdf>.
- R. Davé, D. Anglés-Alcázar, D. Narayanan, Q. Li, M. H. Rafieferantsoa, and S. Appleby, *MNRAS* **486**, 2827 (2019), arXiv:1901.10203 [astro-ph.GA].
- G. Bruzual and S. Charlot, *MNRAS* **344**, 1000 (2003), arXiv:astro-ph/0309134 [astro-ph].
- C. Carr, G. L. Bryan, D. B. Fielding, V. Pandya, and R. S. Somerville, *ApJ* **949**, 21 (2023), arXiv:2211.05115 [astro-ph.GA].
- V. Pandya, D. B. Fielding, G. L. Bryan, C. Carr, R. S. Somerville, J. Stern, C.-A. Faucher-Giguère, Z. Hafen, D. Anglés-Alcázar, and J. C. Forbes, *ApJ* **956**, 118 (2023), arXiv:2211.09755 [astro-ph.GA].
- A. Pillepich, V. Springel, D. Nelson, S. Genel, J. Naiman, R. Pakmor, L. Hernquist, P. Torrey, M. Vogelsberger, R. Weinberger, and F. Marinacci, *MNRAS* **473**, 4077 (2018), arXiv:1703.02970 [astro-ph.GA].
- T. L. Makinen, J. Alsing, and B. D. Wandelt, “Fishnets: Information-optimal, scalable aggregation for sets and graphs,” (2023), arXiv:2310.03812 [cs.LG].
- C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios, *Advances in neural information processing systems* **32** (2019).
- C. Meng, Y. Song, J. Song, and S. Ermon, in *International Conference on Artificial Intelligence and Statistics* (PMLR, 2020) pp. 4336–4345.
- P. Cannon, D. Ward, and S. M. Schmon, arXiv preprint arXiv:2209.01845 (2022).
- R. P. Kelly, D. J. Nott, D. T. Frazier, D. J. Warne, and C. Drovandi, arXiv preprint arXiv:2301.13368 (2023).
- D. Huang, A. Bharti, A. Souza, L. Acerbi, and S. Kaski, arXiv preprint arXiv:2305.15871 (2023).
- F. Leclercq, arXiv e-prints, arXiv:2209.11057 (2022), arXiv:2209.11057 [stat.ME].
- W. Handley, M. Hobson, and A. Lasenby, *Monthly Notices of the Royal Astronomical Society: Letters* **450**, L61 (2015).
- J. D. McEwen, C. G. R. Wallis, M. A. Price, and M. M. Docherty, arXiv (2021), arXiv:2111.12720.
- A. Spurio Mancini, M. Docherty, M. Price, and J. McEwen, *RAS Techniques and Instruments* **2**, 710 (2023).
- N. Jeffrey and B. D. Wandelt, *Machine Learning: Science and Technology* **5**, 015008 (2024).
- L. Masserano, T. Dorigo, R. Izbicki, M. Kuusela, and A. B. Lee, arXiv preprint arXiv:2205.15680 (2022).
- M. Belkin, D. Hsu, S. Ma, and S. Mandal, *Proceedings of the National Academy of Sciences* **116**, 15849 (2019).
- T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (2019) pp. 2623–2631.
- J. Jasche and B. D. Wandelt, *Monthly Notices of the Royal Astronomical Society* **432**, 894 (2013).
- D. de Andres, W. Cui, G. Yepes, M. De Petris, A. Ferragamo, F. De Luca, G. Aversano, and D. Rennehan, *Monthly Notices of the Royal Astronomical Society*, stae071 (2024).
- N. Jeffrey and B. D. Wandelt, arXiv preprint arXiv:2011.05991 (2020).
- R. Legin, M. Ho, P. Lemos, L. Perreault-Levasseur, S. Ho, Y. Hezaveh, and B. Wandelt, *Monthly Notices of the Royal Astronomical Society: Letters* **527**, L173 (2024).
- N. Choustikov, R. Stiskalek, A. Saxena, H. Katz, J. Devrient, and A. Slyz, arXiv preprint arXiv:2405.09720 (2024).
- P. Lemos, S. Sharief, N. Malkin, L. Perreault-Levasseur, and Y. Hezaveh, arXiv preprint arXiv:2402.04355 (2024).
- B. K. Miller, A. Cole, P. Forré, G. Louppe, and C. Weniger, in *35th Conference on Neural Information Processing Systems* (2021) arXiv:2107.01214 [stat.ML].
- L. Sharrock, J. Simons, S. Liu, and M. Beaumont, arXiv preprint arXiv:2210.04872 (2022).
- M. Dax, J. Wildberger, S. Buchholz, S. R. Green, J. H. Macke, and B. Schölkopf, arXiv preprint arXiv:2305.17161 (2023).
- J. Linhart, G. V. Cardoso, A. Gramfort, S. L. Corff, and P. L. Rodrigues, arXiv preprint arXiv:2404.07593 (2024).
- K. Zhang, J. Bloom, and N. Hernitschek, in *NeurIPS 2023 Workshop on Deep Learning and Inverse Problems* (2023).

This paper was built using the Open Journal of Astrophysics L^AT_EX template. The OJA is a journal which

provides fast and easy peer review for new papers in the astro-ph section of the arXiv, making the reviewing process simpler for authors and referees alike. Learn more at <http://astro.theoj.org>.