

Neural Methods for Amortized Inference

Andrew Zammit-Mangion¹, Matthew Sainsbury-Dale^{1,2}, and
Raphaël Huser²

¹School of Mathematics and Applied Statistics, University of Wollongong, Wollongong, New South Wales, Australia; email: azm@uow.edu.au

²Statistics Program, Computer, Electrical and Mathematical Sciences and Engineering Division, King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia

Abstract

Simulation-based methods for statistical inference have evolved dramatically over the past 50 years, keeping pace with technological advancements. The field is undergoing a new revolution as it embraces the representational capacity of neural networks, optimization libraries and graphics processing units for learning complex mappings between data and inferential targets. The resulting tools are amortized, in the sense that, after an initial setup cost, they allow rapid inference through fast feed-forward operations. In this article we review recent progress in the context of point estimation, approximate Bayesian inference, summary-statistic construction, and likelihood approximation. We also cover software, and include a simple illustration to showcase the wide array of tools available for amortized inference and the benefits they offer over Markov chain Monte Carlo methods. The article concludes with an overview of relevant topics and an outlook on future research directions.

Keywords— approximate Bayesian inference; likelihood approximation; likelihood-free inference; neural networks; simulation-based inference; variational Bayes

1 INTRODUCTION

Statistical inference, the process of drawing conclusions on an underlying population from observations, is a cornerstone of evidence-based decision making and scientific discovery. It often relies on a statistical model with unknown or uncertain parameters. Parameter inference consists in estimating and quantifying uncertainty over these model parameters. When the underlying model admits a likelihood function that is analytically and computationally tractable, likelihood-based methods are available that are well-suited for inference. However, in many cases the likelihood function is either unavailable or computationally prohibitive to evaluate, but it is feasible to simulate from the model; this is the case, for example, with many geophysical models (e.g., Siahkoohi et al., 2023), epidemiological models (Fasiolo et al., 2016), cognitive neuroscience models (Fengler et al., 2021), agent-based models (Dyer et al., 2024), and some classes of statistical models like Markov random fields (e.g., Besag, 1986) or models for spatial extremes (e.g., Davison et al., 2012, Huser and Wadsworth, 2022). In such cases, one often resorts to simulation-based techniques to bypass evaluation of the likelihood function. Simulation-based inference requires substantial computing capability, and thus only became viable in the second half of the twentieth century (for early examples, see Hoel and Mitchell 1971 and Ross 1972). The field has evolved much since then, due to the dramatic increase in affordable computing power, and the increased ability to generate, store, and process large datasets.

Several review articles have been published in recent years on simulation-based inference. Cranmer et al. (2020) give a succinct summary of several methods ranging from

Simulation-based inference: Any procedure that makes statistical inference using simulations from a generative model.

approximate Bayesian computation (ABC; Sisson et al., 2018) to density estimation (Diggle and Gratton, 1984) and more recent machine learning approaches, including some of the neural network-based approaches we review here. Blum et al. (2013) and Grazian and Fan (2020) focus on ABC methods, while Drovandi (2018) and Drovandi and Frazier (2022) additionally review indirect inference (Gourieroux et al., 1993), and Bayesian synthetic likelihood (Wood, 2010) approaches to simulation-based inference.

This article differs from existing reviews on simulation-based inference in two ways. First, it only considers methods that incorporate neural networks, which have become state-of-the-art in high-dimensional modeling due to their representational capacity (Hornik, 1989) and due to the increased availability of the software and hardware required to train them. Second, it focuses on neural simulation-based methods that are amortized, that is, that leverage a feed-forward relationship between the data and the inferential target to allow fast inference. The Oxford English Dictionary defines amortization as “the action or practice of gradually writing off the initial cost of an asset”. In the context of neural amortized inference, the initial cost involves training a neural network to learn a complex mapping used for inference (a point estimator, an approximate posterior distribution, an approximate likelihood function, etc.). Once the neural network is trained, parameter inferences can then be made several orders of magnitude faster than classical methods such as Markov chain Monte Carlo (MCMC) sampling. Hence, the initial cost of training the network is “amortized over time”, as the trained network is used over and over again for inferences with new data (see the sidebar titled The Power of Amortization). Amortized inference is also associated with the way humans operate, reusing demanding inferences made in the past to make quick decisions (Gershman and Goodman, 2014).

Neural network: A flexible, highly parameterized nonlinear mapping, constructed using a composition of simple functions connected together in a network.

Training: The process of estimating (or “learning”) parameters in a neural network, typically by minimizing an expected loss.

THE POWER OF AMORTIZATION

Neural networks are ideally suited for situations in which a complex task is repeated, and where the problem justifies a potentially substantial initial outlay. In such settings, one expends a cost to initially train the network with the intention of reaping dividends over time with its repeated use, a strategy known as amortization. The power of amortization is perhaps most clearly exhibited in large language models. Training the BigScience Large Open-science Open-access Multilingual Language Model (BLOOM) required over a million hours of computing on several hundred state-of-the-art graphics processing units (GPUs), each costing many thousands of dollars, for nearly four months (Luccioni et al., 2023). The total estimated consumed energy for training was 433 MWh, which is roughly equivalent to the yearly consumption of 72 Australian households. However, once trained, BLOOM could repeatedly produce polished text outputs (inferences) in response to inputs (data) on a single GPU with a total consumed energy far smaller than that needed to boil water for a cup of tea.

Not all simulation-based inference methods establish a fast-to-evaluate feed-forward relationship between the data and the inferential target. These non-amortized methods require re-optimization or re-simulation every time new data are observed. This makes them unsuitable for situations where the same inferential task must be repeated many times on different datasets, as is often the case in operational settings. For example, NASA’s Orbiting-Carbon-Observatory-2 satellite takes approximately 1,000,000 measurements of light spectra per day. For each spectrum, an estimate of carbon dioxide mole fraction is obtained by computing a Bayesian posterior mode using a forward physics model (Cressie, 2018), requiring an exorbitant amount of computing power when done at scale. Neural networks offer a way to make Bayesian inference from the spectra quickly and accurately at a much lower computational cost (David et al., 2021).

1.1 Article outline

This review provides an introductory high-level roadmap to amortized neural inferential methods. It organizes and categorizes these methodologies to help navigate this relatively new and dynamic field of research. In Section 2 we describe amortization from a decision-theoretic perspective. In Section 3 we discuss (Bayesian) point estimation and methods that yield approximate posterior distributions. In Section 4 we show how neural networks can be used to construct summary statistics, which often feature in amortized statistical inference methods. In Section 5, we review neural methods for amortized likelihood-function and likelihood-to-evidence-ratio approximation. In Section 6, we discuss software for making amortized parameter inference, and we demonstrate its use on a simple model where asymptotically exact inference is possible. The Supplemental Appendix contains a second software example, further reading on topics related to the main subject of the review, and additional figures and tables.

1.2 Notation convention

We use boldface to denote vectors and matrices, $\boldsymbol{\delta}$ and $\boldsymbol{\delta}(\cdot)$ to denote a generic decision and decision rule, respectively, and $\mathbf{X} \in \mathcal{X}$ to denote a generic random vector that takes values in \mathcal{X} . We denote data from a sample space \mathcal{Z} as $\mathbf{Z} \equiv (Z_1, \dots, Z_n)' \in \mathcal{Z} \subseteq \mathbb{R}^n$, and a model parameter vector of interest as $\boldsymbol{\theta} \equiv (\theta_1, \dots, \theta_d)' \in \Theta$, where $\Theta \subseteq \mathbb{R}^d$ is the parameter space. Unless specified otherwise, we assume that all measures admit densities with respect to the Lebesgue measure. To simplify notation, we let the argument of a density function $p(\cdot)$ (which, from hereafter, we simply refer to as a distribution) indicate both the random variable the distribution is associated with and its input argument. In particular, $p(\boldsymbol{\theta})$ is the prior distribution of the parameters, $p(\mathbf{Z})$ is the marginal likelihood (also known as the model evidence), and $p(\boldsymbol{\theta} | \mathbf{Z}) = p(\mathbf{Z} | \boldsymbol{\theta})p(\boldsymbol{\theta})/p(\mathbf{Z})$ is the posterior distribution of $\boldsymbol{\theta} \in \Theta$ given $\mathbf{Z} \in \mathcal{Z}$. We refer to $p(\mathbf{Z} | \boldsymbol{\theta})$ as the likelihood function, whether it is expressing a distribution of \mathbf{Z} for some fixed $\boldsymbol{\theta}$ or a function of $\boldsymbol{\theta}$ for some fixed \mathbf{Z} , with its definition taken from the context in which it is used. We denote an approximate function (e.g., approximate likelihood function or posterior distribution) generically as $q(\cdot)$; we use $\boldsymbol{\kappa}$ to parameterize the approximation when $q(\cdot)$ refers to an approximate posterior distribution, and use $\boldsymbol{\omega}$ when $q(\cdot)$ refers to an approximate likelihood function. We often express these parameters as functions; for example, $\boldsymbol{\kappa}_\gamma(\mathbf{Z})$ is a function parameterized by γ that takes data \mathbf{Z} as input and that outputs parameters characterizing the approximate posterior distribution of $\boldsymbol{\theta}$.

2 AMORTIZATION: A DECISION-THEORETIC PERSPECTIVE

Consider a random vector $\mathbf{X} \in \mathcal{X}$. A decision rule $\boldsymbol{\delta}(\cdot)$ on \mathcal{X} is a function that returns a decision for any given $\mathbf{X} \in \mathcal{X}$. In statistical inference, the most well-known case is where \mathbf{X} is data and the decision rule is a point estimator returning a point estimate from data (Casella and Berger, 2001). Decision rules are typically designed to optimize an objective function, such as a minimum-risk objective function. Optimal decision rules can rarely be expressed in closed form, and often require a computationally expensive optimization procedure to evaluate. *Amortized* optimal decision rules are closed-form expressions that may require considerable modeling and computing effort to construct, but that subsequently yield approximately optimal decisions for any \mathbf{X} with little computing effort. They are central to amortized methods for quickly deriving point estimates, approximate posterior distributions, summary statistics, or approximate likelihood functions.

Finding an optimal decision often involves solving $\boldsymbol{\delta}^* = \inf_{\boldsymbol{\delta}} g(\mathbf{X}, \boldsymbol{\delta})$, where $g(\cdot, \cdot)$ is an objective function to minimize. For example, when $\boldsymbol{\delta}$ is a point estimate and \mathbf{X} is

Point estimator: Any function of the data \mathbf{Z} that returns an estimate of an unknown model parameter $\boldsymbol{\theta}$.

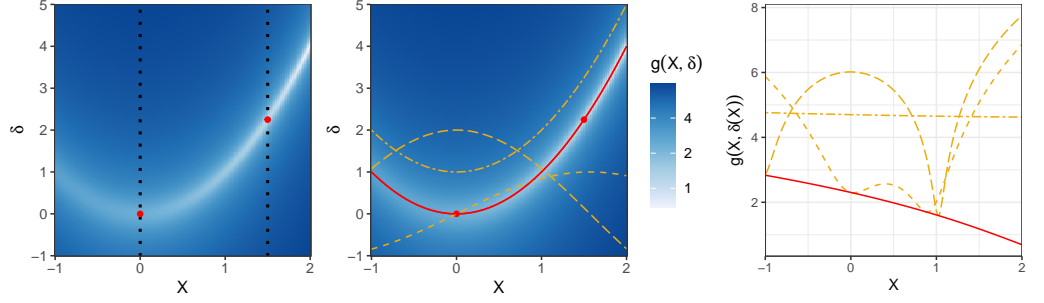


Figure 1: (Left) A hypothetical nonnegative function $g(X, \delta)$. To find the optimal decision δ^* for a given X (shown as red dots for $X = 0$ and $X = 1.5$), $g(X, \delta)$ must be minimized along a slice at X (black dotted lines). (Centre) A function $\delta^*(\cdot)$ (red solid line) that minimizes $g(\cdot, \cdot)$ for any $X \in \mathcal{X}$, whose existence is proved by Brown and Purves (1973), and alternative decision rules (orange dashed lines). If known, or well-approximated, in closed form, $\delta^*(\cdot)$ can be used to quickly make optimal decisions for any $X \in \mathcal{X}$. (Right) The optimal decision rule $\delta^*(\cdot)$ satisfies $g\{X, \delta^*(X)\} = \inf_{\delta} g(X, \delta)$ for all $X \in \mathcal{X}$, and therefore minimizes $\int_{\mathcal{X}} g\{X, \delta(X)\} d\mu(X)$ under any strictly positive measure $\mu(\cdot)$ on \mathcal{X} .

data, a popular choice for $g(\cdot, \cdot)$ is the posterior expected loss (Section 3.1). Repeating this optimization procedure many times for different \mathbf{X} could be time-consuming or computationally prohibitive. Brown and Purves (1973) show that under mild conditions, including $g(\cdot, \cdot)$ being bounded, there exists a decision rule $\delta^*(\cdot)$ that satisfies

$$g\{\mathbf{X}, \delta^*(\mathbf{X})\} = \inf_{\delta} g(\mathbf{X}, \delta), \quad \text{for all } \mathbf{X} \in \mathcal{X}. \quad (1)$$

Equation 1 provides the motivation for amortized methods: Once an accurate closed-form approximation to the decision rule $\delta^*(\cdot)$ is found, computationally-intensive optimization routines are no longer needed. The computational burden is shifted from evaluation of $\delta^*(\mathbf{X})$ for each \mathbf{X} to that of constructing an optimal mapping $\delta^*(\cdot)$.

For illustration, consider the objective function $g(X, \delta) \equiv \log\{(3-X)^2 + 100(\delta - X^2)^2 + 1\}$; see Figure 1, left panel. In a non-amortized setting, the optimal decision δ^* is found by minimizing $g(X, \delta)$ for fixed X . This potentially burdensome optimization problem must be repeated for each X . On the other hand, in an amortized setting, one finds a decision rule with a closed-form expression that returns an approximately optimal decision for any $X \in \mathcal{X}$. For our choice of $g(X, \delta)$ we have an exact closed-form expression for the optimal decision rule: $\delta^*(X) = X^2$; see Figure 1, centre panel. Furthermore, since $g\{X, \delta(X)\}$ is minimized pointwise by $\delta^*(\cdot)$ for each $X \in \mathcal{X}$ (see Figure 1, right panel) and $g(\cdot, \cdot)$ is nonnegative, the optimal decision rule satisfies $\delta^*(\cdot) = \arg \min_{\delta(\cdot)} \int_{\mathcal{X}} g\{X, \delta(X)\} d\mu(X)$ under any strictly positive measure $\mu(\cdot)$ on \mathcal{X} . This property, which we refer to as *average optimality*, is used for finding approximations to optimal decision rules.

A ubiquitous choice for $g(\cdot, \cdot)$ is a Kullback–Leibler (KL) divergence (Kullback and Leibler, 1951) which involves a target distribution $p(\cdot)$ and an approximate distribution $q(\cdot)$. For example, in posterior inference (Section 3), $p(\cdot)$ is a posterior distribution, $q(\cdot)$ is an approximate posterior distribution, and the decision rule $\delta^*(\cdot)$ returns parameters of $q(\cdot)$ from data \mathbf{Z} . In the context of likelihood-function approximation (Section 5.1), $p(\cdot)$ is a likelihood function, $q(\cdot)$ is an approximate (or synthetic) likelihood function, and $\delta^*(\cdot)$ returns parameters used to construct $q(\cdot)$ from model parameters θ . In both contexts, amortization is achieved by constructing approximate closed-form representations of optimal decision rules; we focus on cases where neural networks are used for this approximation.

Kullback–Leibler (KL) divergence: A dissimilarity measure between two distributions. For two densities $p(\cdot)$ and $q(\cdot)$, the forward KL divergence between $p(\cdot)$ and $q(\cdot)$ is $\text{KL}\{p(\mathbf{X}) \parallel q(\mathbf{X})\} \equiv \int p(\mathbf{X}) \log \frac{p(\mathbf{X})}{q(\mathbf{X})} d\mathbf{X}$. We denote the reverse KL divergence by $\text{KL}\{q(\mathbf{X}) \parallel p(\mathbf{X})\}$.

3 NEURAL POSTERIOR INFERENCE

We now review two prominent classes of amortized inferential approaches: neural Bayes estimation (Section 3.1) and methods for neural posterior inference (Section 3.2) that approximate Bayesian posterior distributions via the minimization of an expected KL divergence.

3.1 Neural Bayes Estimators

Here we consider the decision-theoretic framework of Section 2 where \mathbf{X} is data $\mathbf{Z} \in \mathcal{Z}$, the decision δ is an estimate $\theta \in \Theta$, and the decision rule $\delta(\cdot)$ is an estimator $\hat{\theta} : \mathcal{Z} \rightarrow \Theta$. Consider a loss function $L : \Theta \times \Theta \rightarrow \mathbb{R}^{\geq 0}$ and let $g\{\mathbf{Z}, \hat{\theta}(\mathbf{Z})\} = \int_{\Theta} L\{\theta, \hat{\theta}(\mathbf{Z})\} p(\theta | \mathbf{Z}) d\theta$, the posterior expected loss. Then, by average optimality,

$$\hat{\theta}^*(\cdot) = \arg \min_{\hat{\theta}(\cdot)} \mathbb{E}_{\mathbf{Z}} \left[\int_{\Theta} L\{\theta, \hat{\theta}(\mathbf{Z})\} p(\theta | \mathbf{Z}) d\theta \right]. \quad (2)$$

An application of Fubini's theorem to Equation 2 (Robert, 2007, Theorem 2.3.2) and Bayes' rule reveals that $\hat{\theta}^*(\cdot) = \arg \min_{\hat{\theta}(\cdot)} r_B\{\hat{\theta}(\cdot)\}$ where

$$r_B\{\hat{\theta}(\cdot)\} \equiv \int_{\Theta} \int_{\mathcal{Z}} L\{\theta, \hat{\theta}(\mathbf{Z})\} p(\mathbf{Z} | \theta) p(\theta) d\mathbf{Z} d\theta, \quad (3)$$

is the Bayes risk. A minimizer of $r_B\{\hat{\theta}(\cdot)\}$ is said to be a Bayes estimator. Equation 3 can also be derived by considering the approximate posterior distribution $q\{\theta; \hat{\theta}(\mathbf{Z})\} \propto \exp[-L\{\theta; \hat{\theta}(\mathbf{Z})\}]$ and setting $g\{\mathbf{Z}, \hat{\theta}(\mathbf{Z})\} = \text{KL}[p(\theta | \mathbf{Z}) \parallel q\{\theta; \hat{\theta}(\mathbf{Z})\}]$, the forward KL divergence between $p(\cdot | \mathbf{Z})$ and $q\{\cdot; \hat{\theta}(\mathbf{Z})\}$. Bayes estimators are thus a special case of the forward KL divergence minimization methods reviewed in Section 3.2.1.

Assume now that $\hat{\theta}_{\gamma}(\cdot)$ is a neural network flexible enough to approximate the true Bayes estimator arbitrarily well, where γ are the network's parameters. Its architecture is largely determined by the structure of the data: For example, if the data are images, then $\hat{\theta}_{\gamma}(\cdot)$ is typically a convolutional neural network (CNN), which is designed for gridded data. A simulation-based approach to train the network proceeds as follows: In Equation 3, replace the estimator with the neural network; approximate the Bayes risk with a Monte Carlo approximation using N simulations of parameters and corresponding datasets from an underlying model, $\{\{\theta^{(i)}, \mathbf{Z}^{(i)}\} : i = 1, \dots, N\}$; and then solve the empirical risk minimization problem

$$\gamma^* = \arg \min_{\gamma} \sum_{i=1}^N L\{\theta^{(i)}, \hat{\theta}_{\gamma}(\mathbf{Z}^{(i)})\}. \quad (4)$$

Minimization is typically done using stochastic gradient descent in conjunction with automatic differentiation, implemented using machine-learning software libraries (see Section 6). We call the trained network a neural Bayes estimator. See Sainsbury-Dale et al. (2024, Section 2) for more discussion and for an example comparing a neural Bayes estimator to the analytic Bayes estimator for a simple model. A graphical representation of the neural Bayes estimator is given in Figure 2.

Neural networks are extensively used for parameter point estimation. Gerber and Nychka (2021), Lenzi et al. (2023), and Sainsbury-Dale et al. (2024) use CNNs to estimate covariance function parameters in spatial Gaussian process models and models of spatial extremes that are considered computationally difficult to fit. Liu et al. (2020) also estimate the parameters of a Gaussian process, but they adopt a transformer network to cater for realizations of varying dimension and highly parameterized covariance functions. Zammit-Mangion and Wikle (2020) use a CNN with three-dimensional kernels to estimate the dynamical parameters of an advection-diffusion equation, while Rudi et al.

Loss function: A nonnegative function $L(\theta, \hat{\theta})$ which, in point estimation, quantifies the loss incurred when using $\hat{\theta}$ as an estimate of θ .

Neural-network architecture: The specific design of a neural network, including the operations executed in each layer, how the layers are connected, and the number of layers.

Convolutional neural network (CNN): A neural network architecture involving multiple convolutional operations using kernels that are estimated during training.

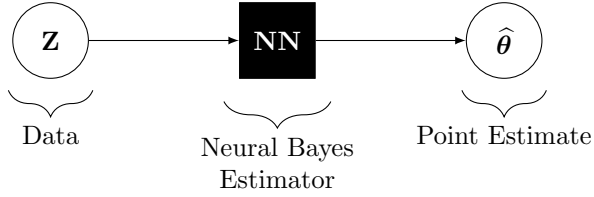


Figure 2: Graphical representation of the neural Bayes estimator that takes data as input and that outputs a parameter point estimate.

(2022) apply a CNN with one-dimensional kernels for parameter estimation with ordinary differential equations. In all cases, a few minutes to a few hours are needed to train the neural networks, but once trained they provide estimates at a small fraction of the time required by classical estimators that require optimization or MCMC for any $\mathbf{Z} \in \mathcal{Z}$ (a speedup of at least a hundredfold is typical). A side benefit of quick parameter estimation is the quantification of uncertainty via parametric or non-parametric bootstrap techniques that require minimal extra computation.

Bayes estimators are functionals of the posterior distribution: Under the squared-error loss $L_{\text{se}}\{\theta, \hat{\theta}(\mathbf{Z})\} \equiv \|\hat{\theta}(\mathbf{Z}) - \theta\|^2$, the Bayes estimator is the posterior mean $\mathbb{E}(\theta \mid \mathbf{Z})$; under the loss $L_{\text{var}}\{\theta_j, \hat{\theta}_j(\mathbf{Z})\} \equiv [\{\theta_j - \mathbb{E}(\theta_j \mid \mathbf{Z})\}^2 - \hat{\theta}_j(\mathbf{Z})]^2$ for some parameter $\theta_j, j = 1, \dots, d$, the Bayes estimator is the posterior variance $\mathbb{V}(\theta_j \mid \mathbf{Z})$ (Fan and Yao, 1998); and under the quantile loss function $L_\rho\{\theta_j, \hat{\theta}_j(\mathbf{Z})\} \equiv \{\hat{\theta}_j(\mathbf{Z}) - \theta_j\}[\mathbb{1}\{\hat{\theta}_j(\mathbf{Z}) > \theta_j\} - \rho]$ the Bayes estimator is the posterior ρ -quantile. Posterior quantiles can be used for fast quantification of posterior uncertainty of θ via the construction of credible intervals. For example, Sainsbury-Dale et al. (2023) obtain posterior medians and 95% credible intervals of Gaussian-process-covariance-function parameters in over 2,000 spatial regions from a million observations of sea-surface temperature in just three minutes on a single GPU. The required time is much less than that required by classical techniques such as non-amortized MCMC or variational inference.

Amortization gap: Error introduced in amortized inference because of incomplete training (e.g., not enough simulations), or because of neural network inflexibility, or both.

The outer expectation in Equation 2 leads to the point estimator $\hat{\theta}^*(\cdot)$ being optimal, in a Bayes sense, for any \mathbf{Z} (see Section 2). However, in practice, there will be some discrepancy between a trained neural network $\hat{\theta}_{\gamma^*}(\cdot)$ and the true Bayes estimator $\hat{\theta}^*(\cdot)$. Any extra bias or variance introduced that makes the trained estimator sub-optimal with respect to the target optimal (in a KL sense) estimator is referred to as the amortization gap (Cremer et al., 2018), and this gap is a consideration for this and all the remaining approaches discussed in this review.

3.2 Approximate Bayesian inference via KL-divergence minimization

Minimizing the KL divergence between $p(\theta \mid \mathbf{Z})$ and an approximate posterior distribution underpins many inference techniques. We focus on amortized versions of the two main techniques: forward KL divergence minimization, and reverse KL divergence minimization (i.e., variational Bayes). These approximate Bayesian methods are often used when asymptotically exact methods such as MCMC are computationally infeasible. The approximate distribution $q(\theta; \kappa)$ has parameters κ that need to be estimated. For example, when $q(\theta; \kappa)$ is chosen to be Gaussian, the parameters $\kappa = (\mu', \text{vech}(\mathbf{L})')'$ comprise a d -dimensional mean parameter μ and the $d(d+1)/2$ non-zero elements of the lower Cholesky factor \mathbf{L} of a covariance matrix, where the half-vectorization operator $\text{vech}(\cdot)$ vectorizes the lower triangular part of its matrix argument.

Variational Bayes: A type of approximate Bayesian inference where the approximate distribution is that which minimizes the reverse KL divergence between the true posterior distribution and its approximation.

Forward and reverse KL minimization approaches both target the true posterior distribution, in the sense that in both cases the KL divergence is zero if and only if $q(\theta)$ is

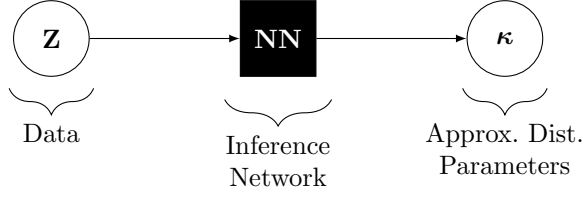


Figure 3: Graphical representation of an inference network that takes data as input and that outputs parameters of an approximate posterior distribution.

identical to $p(\boldsymbol{\theta} \mid \mathbf{Z})$ for $\boldsymbol{\theta} \in \Theta$. However, when $p(\boldsymbol{\theta} \mid \mathbf{Z})$ is not in the class of approximating distributions, the optimal approximate distribution depends on the direction of the KL divergence. As shown by Murphy (2012, Chapter 21), minimizing the reverse KL divergence leads to approximate distributions that are under-dispersed and that tend to concentrate mass on a single mode of the target distribution, whereas minimizing the forward KL divergence leads to ones that are over-dispersed and that cover all modes of the target distribution. Although both approaches are ubiquitous, in simulation-based settings forward KL approaches offer some advantages over their reverse KL counterparts; namely, they are natively likelihood-free and they have a more straightforward implementation.

3.2.1 Forward KL-divergence minimization

We first consider the non-amortized context, where we find the optimal approximate-distribution parameters $\boldsymbol{\kappa}$ by forward KL divergence minimization:

$$\boldsymbol{\kappa}^* = \arg \min_{\boldsymbol{\kappa}} \text{KL}\{p(\boldsymbol{\theta} \mid \mathbf{Z}) \parallel q(\boldsymbol{\theta}; \boldsymbol{\kappa})\}. \quad (5)$$

The optimization problem in Equation 5 is typically computationally expensive to solve even for a single \mathbf{Z} : Solving it for many different datasets is often computationally prohibitive. The learning problem can be amortized as described in Section 2, by treating the decision $\boldsymbol{\delta}$ as the parameter vector $\boldsymbol{\kappa}$ and the decision rule $\boldsymbol{\delta}(\cdot)$ as the function $\boldsymbol{\kappa}(\cdot)$ that maps data to distribution parameters. We then replace $q(\boldsymbol{\theta}; \boldsymbol{\kappa})$ with $q\{\boldsymbol{\theta}; \boldsymbol{\kappa}(\mathbf{Z})\}$ for $\boldsymbol{\theta} \in \Theta, \mathbf{Z} \in \mathcal{Z}$, set $g\{\mathbf{Z}, \boldsymbol{\kappa}(\mathbf{Z})\}$ to be the forward KL divergence between $p(\cdot)$ and $q(\cdot)$, and use average optimality to assert that

$$\boldsymbol{\kappa}^*(\cdot) = \arg \min_{\boldsymbol{\kappa}(\cdot)} \mathbb{E}_{\mathbf{Z}}(\text{KL}[p(\boldsymbol{\theta} \mid \mathbf{Z}) \parallel q\{\boldsymbol{\theta}; \boldsymbol{\kappa}(\mathbf{Z})\}]). \quad (6)$$

In neural amortized inference, one uses a neural network $\boldsymbol{\kappa}_{\boldsymbol{\gamma}}(\cdot)$ for $\boldsymbol{\kappa}(\cdot)$, where $\boldsymbol{\gamma}$ are the neural network parameters. The function $\boldsymbol{\kappa}_{\boldsymbol{\gamma}}(\cdot)$ is often referred to as an inference network, and $\boldsymbol{\gamma}$ is found by minimizing a Monte Carlo approximation to the expected KL divergence in Equation 6. Specifically, the optimization problem becomes

$$\boldsymbol{\gamma}^* = \arg \min_{\boldsymbol{\gamma}} - \sum_{i=1}^N \log q\{\boldsymbol{\theta}^{(i)}; \boldsymbol{\kappa}_{\boldsymbol{\gamma}}(\mathbf{Z}^{(i)})\}, \quad (7)$$

where $\boldsymbol{\theta}^{(i)} \sim p(\boldsymbol{\theta})$ and $\mathbf{Z}^{(i)} \sim p(\mathbf{Z} \mid \boldsymbol{\theta}^{(i)})$. This approach to neural posterior inference has the graphical representation shown in Figure 3.

Chan et al. (2018) let $q\{\cdot; \boldsymbol{\kappa}_{\boldsymbol{\gamma}}(\mathbf{Z})\}$ be Gaussian with mean and precision both functions of the data $\mathbf{Z} \in \mathcal{Z}$, while Maceda et al. (2024) consider a trans-Gaussian approximate posterior distribution. Papamakarios and Murray (2016) go further and let $q\{\cdot; \boldsymbol{\kappa}_{\boldsymbol{\gamma}}(\mathbf{Z})\}$ be a Gaussian mixture. Further flexibility can be achieved by modeling $q\{\cdot; \boldsymbol{\kappa}_{\boldsymbol{\gamma}}(\mathbf{Z})\}$ using a normalizing flow. In such cases, the architecture used to construct $q\{\cdot; \boldsymbol{\kappa}_{\boldsymbol{\gamma}}(\mathbf{Z})\}$ is

Inference network: A neural network whose output can be used to construct an approximate posterior distribution from data \mathbf{Z} or summaries thereof.

Normalizing flow: A sequence of invertible mappings used to establish a flexible family of distributions.

Invertible neural network: A neural network whose architecture constrains it to lie in the space of invertible mappings.

often termed a conditional invertible neural network, since the invertible map constructed using the neural network has parameters determined by \mathbf{Z} ; see, for example, Ardizzone et al. (2019) and Radev et al. (2022). From Equation 7 note that we must be able to evaluate $q\{\boldsymbol{\theta}; \boldsymbol{\kappa}(\mathbf{Z})\}$ for any $\boldsymbol{\theta} \in \Theta$ and $\mathbf{Z} \in \mathcal{Z}$ when training the neural network. This is possible using normalizing flows, which are invertible by construction. However, invertibility can be a restrictive requirement; Pacchiardi and Dutta (2022a) present an approach to circumvent this restriction by instead using a proper scoring rule (Gneiting et al., 2007) in the objective function (Equation 7).

3.2.2 Reverse KL-divergence minimization

Several methods for amortized inference minimize the expected reverse KL divergence between the true posterior distribution and the approximate posterior distribution,

$$\boldsymbol{\kappa}^*(\cdot) = \arg \min_{\boldsymbol{\kappa}(\cdot)} \mathbb{E}_{\mathbf{Z}}(\text{KL}[q\{\boldsymbol{\theta}; \boldsymbol{\kappa}(\mathbf{Z})\} \parallel p(\boldsymbol{\theta} \mid \mathbf{Z})]).$$

In this case, the approximate posterior distribution is referred to as a variational posterior distribution. As in Section 3.2.1, the parameters appearing in the variational posterior distribution are functions of the data. The decision rule $\boldsymbol{\kappa}^*(\cdot)$ minimizes a KL divergence between the true posterior distribution $p(\boldsymbol{\theta} \mid \mathbf{Z})$ and the approximate posterior distribution for every $\mathbf{Z} \in \mathcal{Z}$, and hence yields optimal variational-posterior-distribution parameters for every $\mathbf{Z} \in \mathcal{Z}$. We now replace $\boldsymbol{\kappa}(\cdot)$ with an inference (neural) network $\boldsymbol{\kappa}_{\boldsymbol{\gamma}}(\cdot)$ (Mnih and Gregor, 2014). The neural network parameters $\boldsymbol{\gamma}$ can be optimized by minimizing a Monte Carlo approximation to the expected reverse KL divergence,

$$\boldsymbol{\gamma}^* \approx \arg \min_{\boldsymbol{\gamma}} \sum_{k=1}^K \sum_{j=1}^J \left[\log q\{\boldsymbol{\theta}^{(j)}; \boldsymbol{\kappa}_{\boldsymbol{\gamma}}(\mathbf{Z}^{(k)})\} - \log \{p(\mathbf{Z}^{(k)} \mid \boldsymbol{\theta}^{(j)})p(\boldsymbol{\theta}^{(j)})\} \right], \quad (8)$$

where $\mathbf{Z}^{(k)} \sim p(\mathbf{Z})$, and $\boldsymbol{\theta}^{(j)} \sim q\{\boldsymbol{\theta}; \boldsymbol{\kappa}_{\boldsymbol{\gamma}}(\mathbf{Z}^{(k)})\}$. The objective function in Equation 8 involves samples from both the model and the variational distribution. Since the latter quantity is itself a function of $\boldsymbol{\gamma}$, one generally invokes the so-called reparameterization trick on $q\{\boldsymbol{\theta}; \boldsymbol{\kappa}_{\boldsymbol{\gamma}}(\mathbf{Z}^{(k)})\}$ to ensure that gradients with respect to $\boldsymbol{\gamma}$ can be evaluated (Kingma and Welling, 2013), which are required for training the neural network.

Reparameterization trick: Implicitly defining a parameter-dependent distribution via a generative model through which gradients can be propagated. For example, re-expressing $X \sim \text{Gau}(\mu, \sigma^2)$ as $X = \mu + \sigma\epsilon$, $\epsilon \sim \text{Gau}(0, 1)$.

The concept of an inference network that returns parameters of a variational posterior distribution dates back to Dayan et al. (1995), where $\boldsymbol{\kappa}_{\boldsymbol{\gamma}}(\cdot)$ was called a recognition model. It has since been used in various ways, for example for inference on hyperparameters in Gaussian process models (Rehn, 2022) or on latent variables within deep Gaussian process models (Dai et al., 2015). Rezende and Mohamed (2015) and Kingma et al. (2016) add flexibility to the variational posterior distribution by constructing $q\{\cdot; \boldsymbol{\kappa}_{\boldsymbol{\gamma}}(\mathbf{Z})\}$, $\mathbf{Z} \in \mathcal{Z}$, using a normalizing flow, so that the inference network $\boldsymbol{\kappa}_{\boldsymbol{\gamma}}(\cdot)$ returns both the parameters of the flow’s starting distribution and those governing the flow. Variational auto-encoders (Kingma and Welling, 2013) comprise an inference network in the encoding stage that maps the data directly to parameters of the approximate posterior distribution. More in-depth treatments of amortized variational inference are provided by Zhang et al. (2018), Ganguly et al. (2023), and Margossian and Blei (2023).

Despite their appeal, variational learning networks are limited by the fact that Equation 8 involves the likelihood function $p(\mathbf{Z} \mid \boldsymbol{\theta})$. In some cases, this function is known and tractable; for example, Equation 8 can be used to develop an amortized variational inference framework for solving inverse problems (Goh et al., 2019, Svendsen et al., 2023), in which the conditional distribution of \mathbf{Z} is assumed to be Gaussian with mean equal to the output of a forward physics model that takes $\boldsymbol{\theta}$ as input. Zhang et al. (2023) also adopt amortized variational inference for estimating parameters in a hierarchical model for spatial extremes that has a tractable likelihood function. When the likelihood function is intractable, however, reverse KL-minimization techniques often use neural-network approaches for likelihood approximation; see Section 5.1 for further details.

4 NEURAL SUMMARY STATISTICS

Simulation-based inferential approaches often require pre-specified summary statistics, provided through expert judgement, indirect inference (e.g., Drovandi et al., 2011) or point summaries of the posterior distribution (e.g., Fearnhead and Prangle, 2012). Blum and Francois (2010), Creel (2017), Gerber and Nychka (2021), and Rai et al. (2024), for example, train neural Bayes estimators using hand-crafted summary statistics rather than raw data. In Section 4.1, we review methods that employ neural networks to explicitly construct summary statistics for downstream inference; in Section 4.2, we discuss methods that integrate summary-statistic construction implicitly in an amortized inference framework; and, in Section 4.3, we discuss how the number of summary statistics might be chosen.

4.1 Explicit neural summary statistics

One of the most common neural summary statistics is the neural Bayes estimator. Jiang et al. (2017), for example, use a neural Bayes estimator under squared error loss to obtain the posterior mean quickly for use in ABC, while Dinev and Gutmann (2018) use a neural Bayes estimator in conjunction with the ratio-based likelihood-free inference method of Thomas et al. (2022); similar approaches are also proposed by Creel (2017), Åkesson et al. (2022) and Albert et al. (2022). Beyond point estimators, neural networks can be trained to return other summaries that are highly informative of θ . A common approach is to target sufficiency of the summary statistics $\mathbf{S}(\mathbf{Z})$ by maximizing the mutual information between $\theta \in \Theta$ and $\mathbf{S}(\mathbf{Z}) \in \mathcal{S}$, where \mathcal{S} is the sample space of $\mathbf{S}(\mathbf{Z})$ (see Barnes et al., 2011, and the discussion by Barnes, Filippi and Stumpf in Fearnhead and Prangle, 2012). The mutual information, $\text{MI}\{\theta; \mathbf{S}(\mathbf{Z})\}$, is defined as the KL divergence between the joint distribution $p\{\theta, \mathbf{S}(\mathbf{Z})\}$ and the product of marginals distributions $p(\theta)p\{\mathbf{S}(\mathbf{Z})\}$. Intuitively, θ and $\mathbf{S}(\mathbf{Z})$ are nearly independent when the mutual information is small, so the summary statistic $\mathbf{S}(\mathbf{Z})$ is irrelevant for describing or predicting θ , and conversely when the mutual information is large. Assume now that a neural network is used to construct summary statistics from data. The so-called summary network $\mathbf{S}_\tau(\cdot)$ with network parameters τ is trained by solving

$$\tau^* = \arg \max_{\tau} \text{MI}\{\theta; \mathbf{S}_\tau(\mathbf{Z})\} = \arg \max_{\tau} \text{KL}[p\{\theta, \mathbf{S}_\tau(\mathbf{Z})\} \parallel p(\theta)p\{\mathbf{S}_\tau(\mathbf{Z})\}]. \quad (9)$$

The objective in Equation 9 is difficult to approximate using Monte Carlo techniques since it involves the distribution of the unknown summary statistics. To circumvent this, the mutual information neural estimator (MINE; Belghazi et al., 2018) replaces the KL divergence with its Donsker–Varadhan (1983) representation. Writing $(\theta', \mathbf{S}_\tau(\mathbf{Z})')' \sim p\{\theta, \mathbf{S}_\tau(\mathbf{Z})\}$ and $(\tilde{\theta}', \mathbf{S}_\tau(\tilde{\mathbf{Z}})')' \sim p(\theta)p\{\mathbf{S}_\tau(\mathbf{Z})\}$, we have

$$\text{MI}\{\theta; \mathbf{S}(\mathbf{Z})\} = \sup_{T: \Theta \times \mathcal{S} \rightarrow \mathbb{R}} \mathbb{E}_{(\theta', \mathbf{S}_\tau(\mathbf{Z})')'}[T\{\theta, \mathbf{S}_\tau(\mathbf{Z})\}] - \log \mathbb{E}_{(\tilde{\theta}', \mathbf{S}_\tau(\tilde{\mathbf{Z}})')'}[\exp T\{\tilde{\theta}, \mathbf{S}_\tau(\tilde{\mathbf{Z}})\}].$$

The appeal of this representation is that it does not require the distribution of the summary statistics when approximated using Monte Carlo methods. In MINE, the function $T(\cdot, \cdot)$ is modeled using a neural network $T_\zeta(\cdot, \cdot)$ that is trained in tandem with $\mathbf{S}_\tau(\cdot)$ via

$$(\tau^*, \zeta^*)' = \arg \max_{(\tau', \zeta')'} \frac{1}{N} \sum_{i=1}^N T_\zeta\{\theta^{(i)}, \mathbf{S}_\tau(\mathbf{Z}^{(i)})\} - \log \left(\frac{1}{N} \sum_{i=1}^N \exp[T_\zeta\{\theta^{(i)}, \mathbf{S}_\tau(\mathbf{Z}^{\{\pi(i)\}})\}] \right),$$

where $\theta^{(i)} \sim p(\theta)$, $\mathbf{Z}^{(i)} \sim p(\mathbf{Z} \mid \theta^{(i)})$ and $\pi(\cdot)$ is a random permutation function used to ensure that $(\theta^{(i)'}, \mathbf{S}_\tau(\mathbf{Z}^{\{\pi(i)\}})')' \sim p(\theta)p\{\mathbf{S}_\tau(\mathbf{Z})\}$.

Indirect inference: Fitting an auxiliary model, that is simpler than the target model, to data, and making inference on target parameters from the fitted auxiliary parameters.

Sufficient statistic: A statistic $\mathbf{S}(\mathbf{Z})$ is sufficient for θ if and only if $p(\mathbf{Z} \mid \theta) = h(\mathbf{Z})f\{\mathbf{S}(\mathbf{Z}); \theta\}$ for some nonnegative functions $h(\cdot), f(\cdot)$ (Fisher–Neyman Factorization Theorem).

Hjelm et al. (2019) found that optimization routines were more stable when the Donsker–Varadhan objective function is replaced with one based on the Jensen–Shannon divergence:

$$\text{MI}\{\boldsymbol{\theta}; \mathbf{S}(\mathbf{Z})\} \approx \sup_{T: \Theta \times \mathcal{S} \rightarrow \mathbb{R}} \mathbb{E}_{(\boldsymbol{\theta}', \mathbf{S}_{\boldsymbol{\tau}}(\mathbf{Z}))'}(-\text{sp}[-T\{\boldsymbol{\theta}, \mathbf{S}_{\boldsymbol{\tau}}(\mathbf{Z})\}]) - \mathbb{E}_{(\tilde{\boldsymbol{\theta}}', \mathbf{S}_{\boldsymbol{\tau}}(\tilde{\mathbf{Z}}))'}(\text{sp}[T\{\tilde{\boldsymbol{\theta}}, \mathbf{S}_{\boldsymbol{\tau}}(\tilde{\mathbf{Z}})\}]),$$

where $\text{sp}(z) \equiv \log\{1 + \exp(z)\}$ is the softplus function. The Jensen–Shannon divergence MI estimator was employed to extract summary statistics for use with both classical simulation-based inference methods and neural based inference methods by Chen et al. (2021). Chen et al. (2023) propose to further tame the optimization problem by using a slice technique that effectively breaks down the high-dimensional information objective into many smaller, lower-dimensional ones.

Other neural-network-based approaches to find summary statistics include those of Charnock et al. (2018) and de Castro and Dorigo (2019), who target statistics that maximize the determinant of the Fisher information matrix. Pacchiardi and Dutta (2022b) take yet another approach and fit a general exponential family model in canonical form to simulated data. Their model comprises a summary statistic network $\mathbf{S}_{\boldsymbol{\tau}}(\cdot)$ and a network for the canonical parameters that together form a so-called neural likelihood function. The networks are fitted by minimizing the Fisher divergence between the true and the neural likelihood functions. Although the neural likelihood function is inferred up to a normalizing constant, and can be used with certain MCMC algorithms like the exchange algorithm, the main appeal of the method is that the neural summary statistics extracted are the sufficient statistics of the best (in a Fisher divergence sense) exponential family approximation to the true likelihood function.

4.2 Implicit neural summary statistics

The ability of neural networks to extract relevant summary statistics from data is analogous to their ability to learn features that are useful for predicting an outcome, for example in image recognition; this property is often referred to as representation learning. Feature extraction happens in the early layers of neural networks; similarly one can design networks for amortized inference so that the first few layers extract informative summary statistics.

All methods discussed in Section 3 can incorporate this design by splitting the underlying neural-network architecture into two parts: a summary network for extracting summary statistics from the data, and an inference network that uses these summary statistics as input to output parameter estimates or distributional parameters. The summary network can be trained in tandem with the now-simplified inference network in an end-to-end manner; see, for example, Sainsbury-Dale et al. (2024) for this setup in an estimation setting, Radev et al. (2022) in a forward KL divergence setting, and Wqvist et al. (2021) in a reverse KL divergence setting. The inference network combined with a summary network in an approximate Bayes setting has the graphical representation shown in Figure 4. When a set of hand-crafted summary statistics is known to be informative of the parameter vector, one may use both these and implicitly learned (neural) summary statistics simultaneously (see, e.g., Sainsbury-Dale et al., 2024, Sec. 2.2.1).

4.3 Number of summary statistics

The dimensionality of $\mathbf{S}(\cdot)$, that is, the number of summary statistics, is a design choice. Pacchiardi and Dutta (2022b) set the number equal to the number of model parameters while Chen et al. (2021) make it twice as large. Chen et al. (2023) propose letting $\mathbf{S}(\cdot)$ be reasonably high-dimensional and retaining summary statistics that substantially contribute to mutual information, noting that twice the dimensionality of $\boldsymbol{\theta}$ suffices in most cases. In the context of neural Bayes estimators, Gerber and Nychka (2021) and

Softplus function: The function $\text{sp}(z) \equiv \log\{1 + \exp(z)\}$ commonly used as a smooth approximation to the rectified linear unit, $\text{ReLU}(z) = \max(0, z)$. Asymptotically, $\text{sp}(z) \approx \exp(z)$ as $z \rightarrow -\infty$ and $\text{sp}(z) \approx z$ as $z \rightarrow \infty$.

Fisher divergence: For two densities $p(\cdot)$ and $q(\cdot)$, the Fisher divergence between $p(\cdot)$ and $q(\cdot)$ is $D_F\{p(\mathbf{X})\|q(\mathbf{X})\} \equiv \frac{1}{2} \int p(\mathbf{X}) \|\nabla \log p(\mathbf{X}) - \nabla \log q(\mathbf{X})\|^2 d\mathbf{X}$.

Representation learning: The learning of features or summaries from data that are useful for downstream inference tasks.

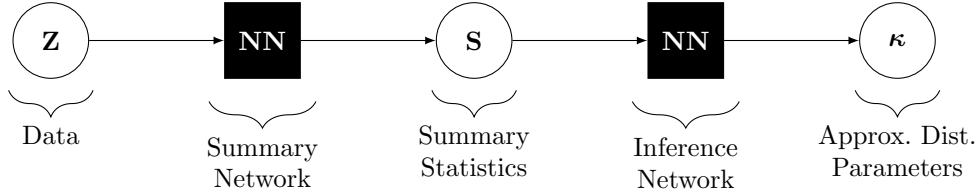


Figure 4: Graphical representation depicting a summary network that takes data as input and that outputs summary statistics, and an inference network that takes the summary statistics as input and that outputs parameters of an approximate posterior distribution.

Sainsbury-Dale et al. (2024) set the number of (implicitly defined) summary statistics to 128, far exceeding the number of model parameters, but irrelevant statistics are down-weighted during training. In practice some experimentation might be needed to determine a suitable number of summary statistics, which should be sufficiently small for training purposes, but large enough for the amortization gap to be within a tolerable range for a given application.

5 NEURAL LIKELIHOOD AND LIKELIHOOD-TO-EVIDENCE RATIO

Section 3 describes methods for amortized neural posterior inference, while Section 4 outlines the role of summary statistics and how they can be automatically constructed from data. In this section we discuss methods for amortized approximation of the likelihood function, $p(\mathbf{Z} \mid \boldsymbol{\theta})$ (Section 5.1), and a closely related quantity that is proportional to the likelihood function, the likelihood-to-evidence ratio, $r(\boldsymbol{\theta}, \mathbf{Z}) = p(\mathbf{Z} \mid \boldsymbol{\theta})/p(\mathbf{Z})$ (Section 5.2).

Amortized neural likelihood and likelihood-to-evidence-ratio approximators have several common motivations. First, the likelihood function is the cornerstone of frequentist inference, while likelihood ratios of the form $p(\mathbf{Z} \mid \boldsymbol{\theta}_0)/p(\mathbf{Z} \mid \boldsymbol{\theta}_1) = r(\boldsymbol{\theta}_0, \mathbf{Z})/r(\boldsymbol{\theta}_1, \mathbf{Z})$ are central to hypothesis testing, model comparison, and naturally appear in the transition probabilities of most standard MCMC algorithms used for Bayesian inference. Second, amortized likelihood and likelihood-to-evidence-ratio approximators enable the straightforward treatment of conditionally (on $\boldsymbol{\theta}$) independent and identically distributed (i.i.d.) replicates since then one has $p(\mathbf{Z}_1, \dots, \mathbf{Z}_m \mid \boldsymbol{\theta}) = \prod_{i=1}^m p(\mathbf{Z}_i \mid \boldsymbol{\theta})$, and the multiple-replicate likelihood-to-evidence ratio is of the form $p(\mathbf{Z}_1, \dots, \mathbf{Z}_m \mid \boldsymbol{\theta})/p(\mathbf{Z}_1, \dots, \mathbf{Z}_m) \propto \prod_{i=1}^m r(\mathbf{Z}_i, \boldsymbol{\theta})$. Hence, an amortized likelihood or likelihood-to-evidence-ratio approximator constructed with single-replicate datasets can be used for frequentist or Bayesian inference with an arbitrary number of conditionally i.i.d. replicates. Third, as these amortized approximators target prior-free quantities, they are ideal for Bayesian inference tasks that require multiple fits of the same model under different prior distributions. However, as we explain below, these methods still require the user to define a proposal distribution $\tilde{p}(\boldsymbol{\theta})$ from which parameters will be sampled during the training stage, which determines the regions of the parameter space where the amortization gap will be lowest.

5.1 Neural Likelihood

We organize this section as follows: In Sections 5.1.1 and 5.1.2 we discuss neural synthetic likelihood and neural full likelihood, respectively; in Section 5.1.3 we provide some comments on the proposal distribution used for training the neural networks; and in Section 5.1.4 we outline an approach that implements neural likelihood and amortized inference simultaneously.

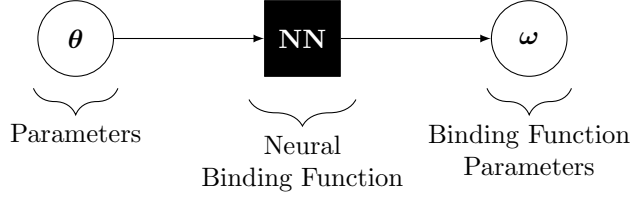


Figure 5: Graphical representation of a neural binding function, a neural network that takes parameters as input and that outputs binding function parameters used to construct a synthetic likelihood function.

5.1.1 Neural synthetic likelihood

A popular method to approximate a likelihood function is the synthetic likelihood framework of Wood (2010). In this framework, one replaces the likelihood function $p(\mathbf{Z} | \theta)$ with a synthetic one of the form $q\{\mathbf{S}(\mathbf{Z}); \omega(\theta)\}$ based on summary statistics $\mathbf{S}(\mathbf{Z})$, where $\omega(\theta)$ is a binding function linking the parameter vector θ to the (approximate) distribution of $\mathbf{S}(\mathbf{Z})$ (e.g., Moores et al., 2015). Here $q\{\mathbf{S}(\mathbf{Z}); \omega(\theta)\}$ refers to a generic (approximate) distribution of $\mathbf{S}(\mathbf{Z})$ evaluated at $\mathbf{S}(\mathbf{Z})$ itself. The summary statistics $\mathbf{S}(\cdot)$ are often modeled as Gaussian with mean parameter $\mu(\theta)$ and covariance matrix $\Sigma(\theta)$. If the binding function $\omega(\theta) = \{\mu(\theta), \Sigma(\theta)\}$ is known, then synthetic likelihood is a form of amortized likelihood approximation, as it allows one to evaluate $p(\cdot | \theta)$ quickly for any θ . When a neural network is used to model the binding function, we obtain a neural synthetic likelihood. In the Gaussian case, we let $q\{\mathbf{S}(\mathbf{Z}); \omega_\eta(\theta)\}$ be a Gaussian function with mean parameter $\mu_\eta(\theta)$ and covariance matrix $\Sigma_\eta(\theta)$, where the neural binding function is $\omega_\eta(\theta) = \{\mu_\eta(\theta), \Sigma_\eta(\theta)\}$, and η are neural network parameters to be estimated. Often, the Gaussianity assumption for $\mathbf{S}(\mathbf{Z})$ is deemed too restrictive, so Radev et al. (2023a) model $q\{\mathbf{S}(\mathbf{Z}); \omega_\eta(\theta)\}$ using normalizing flows.

The binding function is a decision rule ($\delta(\cdot)$ in Section 2) that maps the model parameters θ to the parameters of the synthetic likelihood function. Choosing $g(\cdot, \cdot)$ to be the forward KL divergence between the true likelihood function and the synthetic one, we then apply average optimality to obtain a training objective for the neural binding function,

$$\eta^* = \arg \min_{\eta} \mathbb{E}_{\theta} (\text{KL}[p(\mathbf{Z} | \theta) \parallel q\{\mathbf{S}(\mathbf{Z}); \omega_\eta(\theta)\}]). \quad (10)$$

Equation 10 leads us to consider the empirical risk minimization problem

$$\eta^* = \arg \min_{\eta} - \sum_{i=1}^N \log q\{\mathbf{S}(\mathbf{Z}^{(i)}); \omega_\eta(\theta^{(i)})\}, \quad (11)$$

where $\theta^{(i)} \sim \tilde{p}(\theta)$, $i = 1, \dots, N$, are drawn from some proposal distribution (further discussed below), and $\mathbf{Z}^{(i)} \sim p(\mathbf{Z} | \theta^{(i)})$. The neural binding function has the graphical representation shown in Figure 5.

Once training is complete, the neural synthetic likelihood function can replace the true likelihood function in other inferential techniques, for example in amortized variational inference (Equation 8), as suggested by Wqvist et al. (2021). The resulting framework amortizes the conventional variational-inference-with-synthetic-likelihood approach of Ong et al. (e.g., 2018); we demonstrate its use in Section 6.

5.1.2 Neural full likelihood

While Equation 10 leads to an amortized likelihood approximation framework, replacing the full likelihood with a synthetic counterpart may be undesirable (e.g., when one wishes

Synthetic likelihood function: The distribution of summary statistics (typically assumed to be Gaussian) with parameters that are a function of the underlying model parameters of interest. Often used as a replacement for an intractable likelihood function.

Binding function: Here used to refer to a function mapping the parameters of interest to the parameters of the approximate (synthetic) likelihood function.

to emulate the data \mathbf{Z} for any given θ). One can instead target the full likelihood function, which is obtained by setting $\mathbf{S}(\cdot)$ to be the identity function. Equations 10 and 11 then become

$$\eta^* = \arg \min_{\eta} \mathbb{E}_{\theta} (\text{KL}[p(\mathbf{Z} | \theta) \parallel q\{\mathbf{Z}; \omega_{\eta}(\theta)\}]) = \arg \min_{\eta} \mathbb{E}_{(\theta', \mathbf{Z}')'} [-\log q\{\mathbf{Z}; \omega_{\eta}(\theta)\}], \quad (12)$$

and

$$\eta^* = \arg \min_{\eta} - \sum_{i=1}^N \log q\{\mathbf{Z}^{(i)}; \omega_{\eta}(\theta^{(i)})\}, \quad (13)$$

respectively. Such a framework was considered by Lueckmann et al. (2017) (using Gaussian mixture density networks; see Bishop, 1995, Chapter 5) and Papamakarios et al. (2019) (using masked autoregressive flows).

5.1.3 Choice of proposal distribution

In theory we can use any proposal distribution $\tilde{p}(\theta)$ that is supported over the parameter space to obtain samples for Equation 11 or 13. However, in practice, the chosen proposal distribution is an important consideration because $\tilde{p}(\theta)$ determines the region of the parameter space where the parameters $\{\theta^{(i)}\}$ will be most densely sampled, and thus, where the likelihood approximation will be most accurate. Therefore, the proposal distribution should be sufficiently vague to cover the plausible values of θ , even though, if used in a Bayesian context, it need not be the prior distribution $p(\theta)$. Papamakarios et al. (2019) develop a sequential training scheme aimed at improving simulation efficiency (and thus likelihood approximation accuracy) for specific data \mathbf{Z} .

5.1.4 Joint amortization

Radev et al. (2023a) propose to simultaneously approximate both the posterior distribution and the likelihood function within a single training regime, using two jointly-trained normalizing flows. The method, dubbed jointly amortized neural approximation (JANA), involves extending Equations 6 and 12 to the joint optimization problem,

$$\begin{aligned} (\gamma^{*'}, \tau^{*'}, \eta^{*'})' = \arg \max_{(\gamma', \tau', \eta')'} \mathbb{E}_{(\theta', \mathbf{Z}')'} (\log q[\theta; \kappa_{\gamma}\{\mathbf{S}_{\tau}(\mathbf{Z})\}] + \log q\{\mathbf{Z}; \omega_{\eta}(\theta)\}) \\ - \lambda \cdot (\text{MMD}[p\{\mathbf{S}_{\tau}(\mathbf{Z})\} \parallel \text{Gau}(\mathbf{0}, \mathbf{I})])^2, \quad \lambda > 0, \end{aligned} \quad (14)$$

where $q[\theta; \kappa_{\gamma}\{\mathbf{S}_{\tau}(\mathbf{Z})\}]$ approximates the posterior distribution $p(\theta | \mathbf{Z})$ through the summary network $\mathbf{S}_{\tau}(\mathbf{Z})$, and where $q\{\mathbf{Z}; \omega_{\eta}(\theta)\}$ (or a synthetic likelihood version, based on $\mathbf{S}_{\tau}(\mathbf{Z})$) approximates the likelihood function $p(\mathbf{Z} | \theta)$; the rightmost term on the right-hand side of Equation 14 involves the maximum mean discrepancy (MMD) between the distribution of the summary statistics, $p\{\mathbf{S}_{\tau}(\mathbf{Z})\}$, and the standard multivariate Gaussian distribution, $\text{Gau}(\mathbf{0}, \mathbf{I})$ – this serves as a penalty to impose some probabilistic structure on the summary space and allows for the detection of model misspecification (for further details, see Radev et al., 2023a). Since JANA approximates both the (normalized) posterior distribution and the likelihood function, it also automatically yields an amortized approximation to the marginal likelihood $p(\mathbf{Z}) = p(\mathbf{Z} | \theta)p(\theta)/p(\theta | \mathbf{Z})$, which is key for Bayesian model comparison and selection. Moreover, it also allows one to measure out-of-sample posterior predictive performance via the expected log-predictive distribution. Another joint amortization method, coined the Simformer (Gloeckler et al., 2024), uses a score-based diffusion model (Song et al., 2021) for the joint distribution of θ and \mathbf{Z} , in such a way that samples from any of its conditionals can be easily generated.

Gaussian mixture density network: A Gaussian mixture conditional probability distribution, where the means, variances, and mixing coefficients are the outputs of neural networks that take the conditioning variable as input.

Maximum mean discrepancy: A dissimilarity measure between two distributions. For two densities $p(\cdot)$ and $q(\cdot)$ we denote the discrepancy between $p(\cdot)$ and $q(\cdot)$ as $\text{MMD}\{p(\mathbf{X}) \parallel q(\mathbf{Y})\} = \sup_{g(\cdot) \in \mathcal{G}} [\mathbb{E}_{\mathbf{X}}\{g(\mathbf{X})\} - \mathbb{E}_{\mathbf{Y}}\{g(\mathbf{Y})\}]$, for $\mathbf{X} \sim p(\cdot)$, $\mathbf{Y} \sim q(\cdot)$ and a suitable class of real-valued functions \mathcal{G} (Gretton et al., 2012).

5.2 Neural Likelihood-to-Evidence Ratio

In Section 5.2.1 we outline the general framework for approximating the likelihood-to-evidence ratio, while in Section 5.2.2 we discuss variants that safeguard against overoptimistic inferences and that aim to improve computational efficiency.

5.2.1 General framework

Several approaches to likelihood-free inference are based on ratio approximation (Gutmann and Hyvärinen, 2012). Some approaches (e.g., Cranmer et al., 2015, Baldi et al., 2016) target ratios of the form $p(\mathbf{Z} | \boldsymbol{\theta})/p(\mathbf{Z} | \boldsymbol{\theta}_{\text{ref}})$ for some arbitrary but fixed reference parameter $\boldsymbol{\theta}_{\text{ref}}$. In this review we focus on a particular formulation that obviates the need for a reference parameter by instead targeting the likelihood-to-evidence ratio (Hermans et al., 2020, Thomas et al., 2022),

$$r(\boldsymbol{\theta}, \mathbf{Z}) = p(\mathbf{Z} | \boldsymbol{\theta})/p(\mathbf{Z}). \quad (15)$$

This ratio can be approximated by solving a relatively straightforward binary classification problem, as we now show. Consider a binary classifier $c(\boldsymbol{\theta}, \mathbf{Z})$ that distinguishes dependent parameter-data pairs $(\boldsymbol{\theta}', \mathbf{Z}')' \sim p(\boldsymbol{\theta}, \mathbf{Z} | Y = 1) = p(\boldsymbol{\theta}, \mathbf{Z})$ from independent parameter-data pairs $(\tilde{\boldsymbol{\theta}}', \tilde{\mathbf{Z}}')' \sim p(\boldsymbol{\theta}, \mathbf{Z} | Y = 0) = p(\boldsymbol{\theta})p(\mathbf{Z})$, where Y denotes the class label and where the classes are balanced. Then, we define the optimal classifier $c^*(\cdot, \cdot)$ as that which minimizes the Bayes risk under binary cross-entropy loss, $L_{\text{bce}}(\cdot, \cdot)$,

$$\begin{aligned} c^*(\cdot, \cdot) &\equiv \arg \min_{c(\cdot, \cdot)} \sum_{y \in \{0,1\}} \Pr(Y = y) \int_{\Theta} \int_{\mathcal{Z}} p(\boldsymbol{\theta}, \mathbf{Z} | Y = y) L_{\text{bce}}\{y, c(\boldsymbol{\theta}, \mathbf{Z})\} d\mathbf{Z} d\boldsymbol{\theta} \\ &= \arg \max_{c(\cdot, \cdot)} \sum_{y \in \{0,1\}} \int_{\Theta} \int_{\mathcal{Z}} p(\boldsymbol{\theta}, \mathbf{Z} | Y = y) [y \log\{c(\boldsymbol{\theta}, \mathbf{Z})\} + (1 - y) \log\{1 - c(\boldsymbol{\theta}, \mathbf{Z})\}] d\mathbf{Z} d\boldsymbol{\theta} \\ &= \arg \max_{c(\cdot, \cdot)} \int_{\Theta} \int_{\mathcal{Z}} p(\boldsymbol{\theta}, \mathbf{Z}) \log\{c(\boldsymbol{\theta}, \mathbf{Z})\} d\mathbf{Z} d\boldsymbol{\theta} + \int_{\Theta} \int_{\mathcal{Z}} p(\boldsymbol{\theta})p(\mathbf{Z}) \log\{1 - c(\boldsymbol{\theta}, \mathbf{Z})\} d\mathbf{Z} d\boldsymbol{\theta} \\ &= \arg \max_{c(\cdot, \cdot)} \left[\mathbb{E}_{(\boldsymbol{\theta}', \mathbf{Z}')} \log\{c(\boldsymbol{\theta}, \mathbf{Z})\} + \mathbb{E}_{(\tilde{\boldsymbol{\theta}}', \tilde{\mathbf{Z}}')} \log\{1 - c(\tilde{\boldsymbol{\theta}}, \tilde{\mathbf{Z}})\} \right], \end{aligned} \quad (16)$$

where $\Pr(Y = y)$, $y \in \{0, 1\}$, denotes the class probability. It can be shown that $c^*(\boldsymbol{\theta}, \mathbf{Z}) = p(\boldsymbol{\theta}, \mathbf{Z})\{p(\boldsymbol{\theta}, \mathbf{Z}) + p(\boldsymbol{\theta})p(\mathbf{Z})\}^{-1}$ (e.g., Hermans et al., 2020, App. B) and, hence,

$$r(\boldsymbol{\theta}, \mathbf{Z}) = \frac{c^*(\boldsymbol{\theta}, \mathbf{Z})}{1 - c^*(\boldsymbol{\theta}, \mathbf{Z})}, \quad \boldsymbol{\theta} \in \Theta, \mathbf{Z} \in \mathcal{Z}. \quad (17)$$

In the typical case that $c^*(\cdot, \cdot)$ is unavailable, it can be approximated by adopting a flexible parametric representation for $c(\cdot, \cdot)$ and maximizing a Monte Carlo approximation of the objective function in Equation 16. Specifically, let $c_{\boldsymbol{\gamma}}(\cdot, \cdot)$ denote a binary classifier parameterized by $\boldsymbol{\gamma}$, typically a neural network with a sigmoid output activation function, although other representations are possible (e.g., a logistic regression based on user-specified or learned summary statistics; Dinev and Gutmann, 2018, Thomas et al., 2022). Then, the Bayes classifier may be approximated by $c_{\boldsymbol{\gamma}^*}(\cdot, \cdot)$, where

$$\boldsymbol{\gamma}^* = \arg \max_{\boldsymbol{\gamma}} \sum_{i=1}^N \left[\log\{c_{\boldsymbol{\gamma}}(\boldsymbol{\theta}^{(i)}, \mathbf{Z}^{(i)})\} + \log\{1 - c_{\boldsymbol{\gamma}}(\boldsymbol{\theta}^{(i)}, \mathbf{Z}^{\{\pi(i)\}})\} \right], \quad (18)$$

where $\boldsymbol{\theta}^{(i)} \sim p(\boldsymbol{\theta})$, $\mathbf{Z}^{(i)} \sim p(\mathbf{Z} | \boldsymbol{\theta}^{(i)})$ and $\pi(\cdot)$ is a random permutation of $\{1, \dots, N\}$. Figure 6 demonstrates this learning task on the simple model $Z | \theta \sim \text{Gau}(\theta, \theta^2)$, $\theta \sim \text{Unif}(0, 1)$, for $c_{\boldsymbol{\gamma}}(\cdot, \cdot)$ a fully-connected neural network with six layers, each of width 64.

Once trained, the classifier $c_{\boldsymbol{\gamma}^*}(\cdot, \cdot)$ may be used to quickly approximate the likelihood-to-evidence ratio via Equation 17, as summarized in the graphical representation shown

Balanced classification task:

A classification task is said to be balanced if the prior class probabilities are equal.

Binary cross-entropy (log)

loss: The loss $L_{\text{bce}}(y, c) = -y \log(c) - (1 - y) \log(1 - c)$ (i.e., the negative Bernoulli log-likelihood), a measure of the difference between the true class $y \in \{0, 1\}$ and an estimated class probability $c \in (0, 1)$.

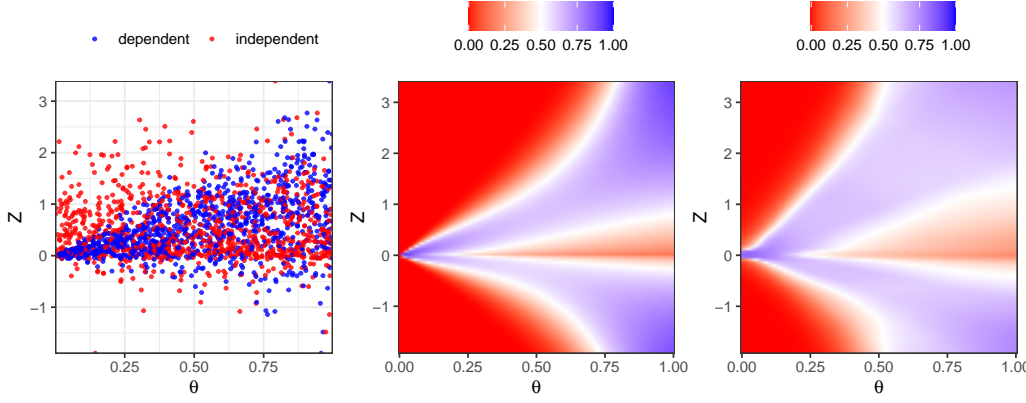


Figure 6: Illustration of amortized likelihood-to-evidence ratio approximation using the model $Z \mid \theta \sim \text{Gau}(\theta, \theta^2)$, $\theta \sim \text{Unif}(0, 1)$. (Left) Samples of dependent pairs $\{\theta, Z\} \sim p(\theta, Z)$ (blue) and independent pairs $\{\theta, \tilde{Z}\} \sim p(\theta)p(Z)$ (red). (Centre) Class probabilities from the exact Bayes classifier $c^*(\theta, Z) = p(\theta, Z)\{p(\theta, Z) + p(\theta)p(Z)\}^{-1}$, linked to the likelihood-to-evidence ratio via the relation $r(\theta, Z) = c^*(\theta, Z)\{1 - c^*(\theta, Z)\}^{-1}$. (Right) Class probabilities from an amortized neural approximation $c_{\gamma^*}(\cdot, \cdot)$ of the Bayes classifier.

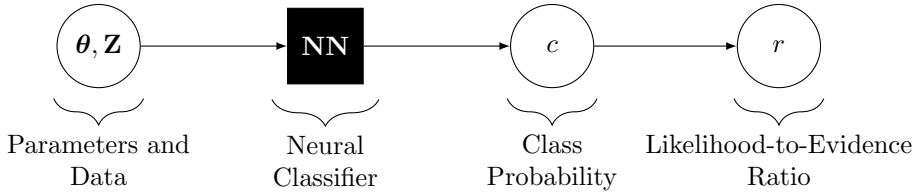


Figure 7: Graphical representation of a neural classifier that takes both data and parameters as input and that outputs a class probability that is then used to approximate the likelihood-to-evidence ratio.

in Figure 7. Inference based on an amortized approximate likelihood-to-evidence ratio may proceed as discussed in the introduction to Section 5, namely, in a frequentist setting via maximum likelihood estimation and likelihood ratios (e.g., Walchessen et al., 2023), and in a Bayesian setting by facilitating the computation of transition probabilities in Hamiltonian Monte Carlo and MCMC algorithms (e.g., Hermans et al., 2020, Begy and Schikuta, 2021). Furthermore, an approximate posterior distribution can be obtained via the identity $p(\boldsymbol{\theta} \mid \mathbf{Z}) = p(\boldsymbol{\theta})r(\boldsymbol{\theta}, \mathbf{Z})$, and sampled from using standard sampling techniques (e.g., Thomas et al. 2022; see also Section 6.2). Finally, if one also has an amortized likelihood approximator (Section 5.1), then one may approximate the model evidence via $p(\mathbf{Z}) = p(\mathbf{Z} \mid \boldsymbol{\theta})/r(\boldsymbol{\theta}, \mathbf{Z})$.

5.2.2 Variants of the target ratio

Several variants of the learning task described above have been investigated, with a variety of different aims. One is that of safeguarding against over-optimistic inferences when the true Bayes classifier $c^*(\cdot, \cdot)$ cannot be well approximated by the neural classifier $c_{\gamma}(\cdot, \cdot)$; in this situation it is widely acknowledged that one should err on the side of caution and make $c_{\gamma}(\cdot, \cdot)$ conservative (Hermans et al., 2022). Delaunoy et al. (2022) propose a way

to do this by encouraging $c_\gamma(\cdot, \cdot)$ to satisfy the balance condition

$$\mathbb{E}_{(\boldsymbol{\theta}', \mathbf{Z}')'} \{c_\gamma(\boldsymbol{\theta}, \mathbf{Z})\} + \mathbb{E}_{(\tilde{\boldsymbol{\theta}}', \tilde{\mathbf{Z}}')'} \{c_\gamma(\tilde{\boldsymbol{\theta}}, \tilde{\mathbf{Z}})\} = 1; \quad (19)$$

it is straightforward to see that the Bayes classifier under equal class probability, $c^*(\cdot, \cdot)$, satisfies the condition. One can show that any classifier satisfying Equation 19 also satisfies

$$\mathbb{E}_{(\boldsymbol{\theta}', \mathbf{Z}')'} \left\{ \frac{c^*(\boldsymbol{\theta}, \mathbf{Z})}{c_\gamma(\boldsymbol{\theta}, \mathbf{Z})} \right\} \geq 1, \quad \text{and} \quad \mathbb{E}_{(\tilde{\boldsymbol{\theta}}', \tilde{\mathbf{Z}}')'} \left\{ \frac{1 - c^*(\tilde{\boldsymbol{\theta}}, \tilde{\mathbf{Z}})}{1 - c_\gamma(\tilde{\boldsymbol{\theta}}, \tilde{\mathbf{Z}})} \right\} \geq 1,$$

which informally means that the classifier tends to produce class-probability estimates that are smaller (i.e., less confident and more conservative) than those produced by the (optimal) Bayes classifier. To encourage the trained estimator to satisfy the balancing condition, and hence be conservative, one may append to the objective function in Equation 16 the penalty term

$$\lambda \left[\mathbb{E}_{(\boldsymbol{\theta}', \mathbf{Z}')'} \{c(\boldsymbol{\theta}, \mathbf{Z})\} + \mathbb{E}_{(\tilde{\boldsymbol{\theta}}', \tilde{\mathbf{Z}}')'} \{c(\tilde{\boldsymbol{\theta}}, \tilde{\mathbf{Z}})\} - 1 \right]^2, \quad \lambda > 0, \quad (20)$$

which equals zero when the balancing condition is satisfied. Since the Bayes classifier with equal class probabilities is balanced, this penalty is only relevant when the Bayes classifier cannot be approximated well using the neural network.

Another research focus is improving computational efficiency for amortized ratio approximators. For example, approximating the likelihood ratio $\tilde{r}(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \mathbf{Z}) \equiv p(\mathbf{Z} | \boldsymbol{\theta}_0)/p(\mathbf{Z} | \boldsymbol{\theta}_1) = r(\boldsymbol{\theta}_0, \mathbf{Z})/r(\boldsymbol{\theta}_1, \mathbf{Z})$ requires two forward passes through a neural classifier $c_{\gamma^*}(\cdot, \cdot)$. Recently, however, Cobb et al. (2023) proposed to construct an amortized approximation of $\tilde{r}(\cdot, \cdot, \cdot)$ using a single neural network that takes two parameter vectors as input. This only requires one forward pass, improving computational efficiency, though this benefit may be offset if a larger neural network is needed to learn the more complicated mapping. Finally, several approaches have been developed to construct amortized ratio approximators for a subset of $\boldsymbol{\theta}$. These marginal methods include introducing a binary mask as input to encode the desired subset of parameters (Rozet and Louppe, 2021), and constructing separate approximators for each subset of interest (Miller et al., 2021, 2022).

6 SOFTWARE AND EXAMPLE

Although amortized neural inference is a relatively new field, many easy-to-use software packages are publicly available. We outline these in Section 6.1 and demonstrate their application through a simple example in Section 6.2 where the model parameters are easily inferred using MCMC. An additional example with a model unsuitable for MCMC is given in Section S1 of the Supplemental Appendix. Code for these two examples is available at https://github.com/andrewzm/Amortized_Neural_Inference_Review/.

6.1 Software

Most publicly available software is written in **Python**, and leverages the deep-learning libraries **TensorFlow** (Abadi et al., 2016) and **PyTorch** (Paszke et al., 2019). Software is also available in the numerical programming language **Julia** and its deep-learning library **Flux** (Innes, 2018). There are R interfaces to several of these libraries written in other languages. Software evolves quickly; the following description is based on availability and functionality as of early 2024.

The package **NeuralEstimators** (Sainsbury-Dale et al., 2024) helps construct neural Bayes estimators (Section 3.1) using arbitrary loss functions, and likelihood-to-evidence ratio approximators (Section 5.2). The package is written in **Julia**, leverages the deep-learning library **Flux**, and is accompanied by a user-friendly R interface. The package is

designed to deal with exchangeable data, such as independent realizations from a spatial process model. The package also implements methods for bootstrap-based uncertainty quantification and for handling censored (Richards et al., 2023) and missing (Wang et al., 2024) data.

The package **sbi** (Tejero-Cantero et al., 2020), short for simulation-based inference, is a **Python** package built on **PyTorch** that provides methods for targeting the posterior distribution (Section 3.2.1), the likelihood function (Section 5.1), or the likelihood-to-evidence ratio (Section 5.2), with posterior inference using the likelihood or likelihood-to-evidence ratio facilitated with MCMC sampling, rejection sampling, or (non-amortized) variational inference. It implements both amortized and sequential methods, the latter aimed at improving simulation efficiency.

The **PyTorch**-based package **LAMPE** (Rozet et al., 2021), short for likelihood-free amortized posterior estimation, focuses on amortized methods for approximating the posterior distribution (Section 3.2.1) or the likelihood-to-evidence ratio (Section 5.2), with posterior inference using the likelihood-to-evidence ratio facilitated with MCMC or nested sampling. The package allows training data to be stored on disk and dynamically loaded on demand, instead of being cached in memory; this technique facilitates the use of very large datasets.

The **TensorFlow**-based package **BayesFlow** (Radev et al., 2023b) implements methods for approximating the posterior distribution (Section 3.2.1) and the likelihood function (Section 5.1), possibly jointly (Radev et al., 2023a); detecting model misspecification (Schmitt et al., 2024a,b); and performing amortized model comparisons via posterior model probabilities or Bayes factors. Recently, neural Bayes estimators (Section 3.1) have also been incorporated into the package. The package is well documented and provides a user-friendly application programming interface.

The **PyTorch**-based package **swyft** (Miller et al., 2022) implements methods for estimating likelihood-to-evidence ratios for subsets of the parameter vector, using amortized and sequential training algorithms. To facilitate the use of datasets that are too large to fit in memory, the package also allows data to be stored on disk and dynamically loaded.

The above software packages provide the tools necessary to easily train the required neural networks from scratch, but are general-purpose. Future amortized inference tools may be bundled with packages primarily designed around a specific modeling framework. For instance, a package that implements a particular model could include a pre-trained neural network, or several such neural networks, designed to make fast inferences for that model. This could be a natural evolution since there are incentives for the developers of such packages to make inference straightforward for increased accessibility and uptake by end users.

6.2 Example

Protocols for benchmarking amortized inference methods are still in their infancy (Lueckmann et al., 2021). To showcase a few techniques discussed in this review, here we consider a model with only one unknown parameter, where the neural networks are straightforward to train without the need for high-end GPUs, and for which MCMC is straightforward to implement for comparison. We consider a spatial Gaussian process with exponential covariance function with unit variance and unknown length scale $\theta > 0$. We assume that data \mathbf{Z} are on a 16×16 gridding, \mathcal{D}^G , of the unit square $\mathcal{D} = [0, 1]^2$. The process model is therefore given by $\mathbf{Z} \mid \theta \sim \text{Gau}\{\mathbf{0}, \Sigma(\theta)\}$, where $\Sigma \equiv (\exp(-\|\mathbf{s}_i - \mathbf{s}_j\|/\theta) : \mathbf{s}_i, \mathbf{s}_j \in \mathcal{D}^G)$, and we let $\theta \sim \text{Unif}(0, 0.6)$. We train the neural networks on 160,000 draws from $p(\theta)$ and $p(\mathbf{Z} \mid \theta)$, and test all methods on 1,000 independent draws.

The techniques we consider, their acronyms, and their software implementations are outlined in Table S1 in the Supplemental Appendix. The gold standard is provided by a (non-amortized) Metropolis–Hastings algorithm, **MCMC**, run on the 1,000 test datasets. We implement a neural Bayes estimator, **NBE** (Section 3.1), using **NeuralEstimators**,

Censored data: Data whose precise values are not known, but which are known to lie in some interval. Multivariate censored data often lead to intractable likelihood functions.

Missing data: Data that have not been observed, and that one often must predict or impute.

Pre-trained neural network:

A neural network trained for a generic task, which is used as a starting network when training for specialized tasks (e.g., in our context, for a slightly different model or prior distribution).

Table 1: Evaluation of the methods in Section 6.2 on test data using the root median squared prediction error (RMSPE), the median 90% interval score (MIS90), the median continuous-ranked probability score (MCRPS), and the 90% empirical coverage (COV90). We use the median instead of the mean since the distribution of the metrics is highly skewed; see Figure S3 in the Supplemental Appendix. All scores excluding COV90 (which should be close to 90%) are negatively oriented (lower is better). MCRPS is not available with our implementation of the NBE.

	MCMC	NBE	fKL	rKL1	rKL2	rKL3	NRE
RMSPE ($\times 10^2$)	1.50	1.56	1.50	1.64	1.88	1.78	1.70
MIS90 ($\times 10^2$)	9.55	10.62	9.89	9.80	11.36	9.07	10.99
MCRPS ($\times 10^2$)	1.41	NA	1.46	1.36	1.61	1.37	1.59
COV90 (%)	89.60	87.70	90.60	77.70	87.10	81.60	91.30

targeting the posterior mean, the posterior 5th percentile and the posterior 95th percentile. We implement amortized posterior inference via forward KL minimization, fKL (Section 3.2.1), using **BayesFlow** with a normalizing flow model for the posterior distribution constructed using affine coupling blocks. In order to ensure that the approximate posterior densities are zero outside the prior support of $[0, 0.6]$, we make inference on $\tilde{\theta} \equiv \Phi^{-1}(\theta/0.6) \sim \text{Gau}(0, 1)$, and obtain samples from the posterior distribution of θ through the inverse $\theta = 0.6\Phi(\tilde{\theta})$, where $\Phi(\cdot)$ is the standard normal cumulative distribution function.

Affine coupling blocks:

An invertible transformation, where the input is partitioned into two blocks. One of the blocks undergoes an affine transformation that depends on the other block.

We implement three types of reverse-KL methods using **TensorFlow**: **rKL1**, **rKL2**, and **rKL3**, which differ in the likelihood function they use. All three target an approximate Gaussian posterior distribution of $\tilde{\theta} \equiv \log(\theta/(0.6 - \theta))$; samples from the approximate (non-Gaussian) posterior distribution of θ are obtained by back-transforming samples drawn from the approximate Gaussian posterior distribution of $\tilde{\theta}$. The first variant, **rKL1**, uses the true likelihood function (Section 3.2.2); the second, **rKL2**, uses a synthetic one constructed with an “expert” summary statistic, in this case given by the mean of the squared differences between neighboring pixels in \mathbf{Z} (Section 5.1.1); and the third, **rKL3**, uses a synthetic one constructed using a summary statistic found by maximizing mutual information (Section 4.1). We also consider the neural ratio estimation method, **NRE**, implemented in the package **sbi** (Section 5.2); we use the amortized ratio to quickly evaluate the posterior distribution on a fine gridding of the parameter space, from which we then draw samples. For all approaches we use similar architectures, largely based on a two-layer CNN. Each neural network took a few minutes to train using the CPU of a standard laptop.

We summarize results in Figure 8 and Table 1, where we assess the methods on the test data using scoring rules (Gneiting and Raftery, 2007) and 90% empirical coverages (e.g., Hermans et al., 2022). Figure 8a shows the summary statistics for **rKL2** and **rKL3**, along with the inferred mean $\mu_{\tau^*}(\theta)$ and the $2\sigma_{\tau^*}(\theta)$ interval used to construct the synthetic likelihood. The expert summary statistic is non-linear and, for large θ , a broad range of parameter values lead to a small expert summary statistic. The statistic constructed by maximizing the mutual information is largely linear over the entire support of $p(\theta)$; as seen from Table 1 this leads to slightly better performance. Figure 8b shows the results from applying the methods to three (test) spatial fields, \mathbf{Z}_1 , \mathbf{Z}_2 , and \mathbf{Z}_3 . All methods perform well, with the posterior variance increasing with θ as expected. Differences between the approaches are mostly evident for larger θ , where inferences are more uncertain. Figure 8c plots the posterior means against the true values; all methods again perform as expected, with lower variance for small θ and large variance for large θ , where the length scale is large in comparison to the size of the spatial domain.

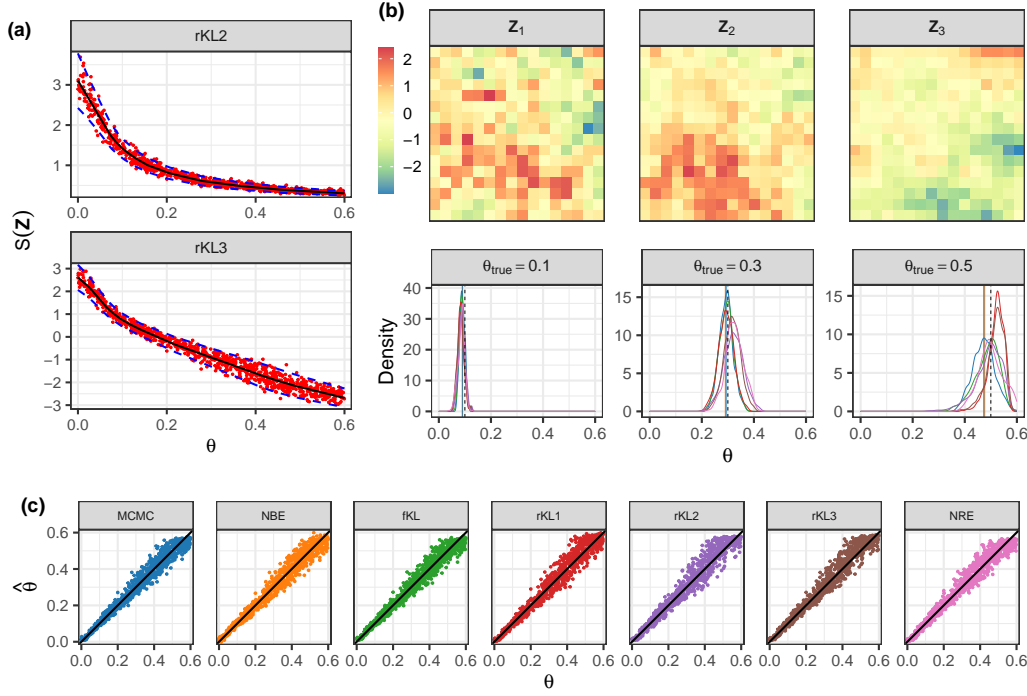


Figure 8: Results from the illustration of Section 6.2. Panel (a) plots the summary statistics of the test data (red), together with the mean (solid black line) and 2σ bands constructed from the fitted neural binding functions. The top sub-panel shows the expert summary statistic (used in **rKL2**), while the bottom sub-panel shows the statistic constructed by maximizing mutual information (used in **rKL3**). Panel (b) shows the results from all methods on three test datasets. The top sub-panels show the data, while the bottom sub-panels show estimates (solid vertical lines, **MCMC** posterior mean and **NBE**, which largely overlap) and the inferred posterior distributions (solid lines, all methods except **NBE**), as well as the true parameter value (dashed line). Panel (c) plots parameter point estimates (in this case the posterior mean) against the true parameter values in the test datasets for all methods. The colours of the lines in panel (b) correspond to the colours used for the different methods in panel (c).

As seen from Table 1, all methods perform slightly worse overall than MCMC due to the amortization gap (NBE, fKL, NRE), the use of an inflexible approximation to the posterior distribution (the rKL variants), or both. However, the significance of amortization lies in the speed with which these inferences can be obtained, and the general class of models the neural methods apply to. In this example, one run of MCMC required around one minute of compute time to generate 24,000 samples (which we thinned down to 1,000), while the neural methods only required a few dozen milliseconds each to yield approximate posterior inferences.

Continuous ranked probability score: Given a forecast distribution F and an observation y , the continuous ranked probability score is $\text{CRPS}(F, y) \equiv \int_{-\infty}^{\infty} \{F(x) - \mathbb{1}(y \leq x)\}^2 dx$.

Interval score: Given the lower and upper quantiles, y_l and y_u , of a forecast distribution, and an observation y , the $(1 - \alpha) \times 100\%$ interval score is $\text{IS}_{\alpha}(y_l, y_u; y) \equiv (y_u - y_l) + \frac{2}{\alpha}(y_l - y)\mathbb{1}(y < y_l) + \frac{2}{\alpha}(y - y_u)\mathbb{1}(y > y_u)$.

7 EPILOGUE

Amortized neural inference methods are new, but they have already shown enormous potential for accurate and fast statistical inference in a wide range of applications with models that either are defined explicitly but are computationally intractable, or that are defined implicitly through a stochastic generator. The amortized nature of the methods presented in this review article, enabled by neural networks for quick evaluation, opens a new era in statistical inference, in which one can construct and share pre-trained inference tools for off-the-shelf use with new data at negligible cost. The field is rapidly expanding and, to preserve the review’s focus, many relevant topics and background materials were not discussed in depth. For further reading on related subjects, see Section S2 of the Supplemental Appendix.

Several challenges remain for future research. From a theoretical viewpoint, we need a better understanding of the asymptotic properties of neural inference tools (e.g., consistency, rate of convergence based on network architecture and training set size) to guide their design and establish rigorous implementation strategies. From a methodological viewpoint, several avenues remain unexplored. For example, recovering distributions of random effects in hierarchical models could be achieved by combining amortized methods for latent variable inference (e.g., Liu and Liu, 2019) with the parameter inference methods discussed in this review. Additionally, some methods like ABC can also be amortized (e.g., Mestdagh et al., 2019), and it is not yet clear what advantages these have, if any, over the approaches discussed in this review. More traditional branches of statistics that can benefit from these advances include analyses of survey data, crop yield, and experimental design. Amortized neural inference approaches also enable online frequentist or Bayesian inference where data arrive sequentially, and where parameters need to be tracked. We anticipate that several fields in statistics will soon be impacted by this relatively new enabling technology.

DISCLOSURE STATEMENT

The authors are not aware of any affiliations, memberships, funding, or financial holdings that might be perceived as affecting the objectivity of this review.

ACKNOWLEDGEMENTS

All authors were supported by the King Abdullah University of Science and Technology (KAUST) Opportunity Fund Program ORFS-2023-OFP-5550.2. This material is based upon work supported by the Air Force Office of Scientific Research under award number FA2386-23-1-4100 (A.Z.-M.). M.S.-D.’s research was additionally supported by an Australian Government Research Training Program Scholarship, a 2022 Statistical Society of Australia (SSA) top-up scholarship, the KAUST Office of Sponsored Research (OSR) under Award No. OSR-CRG2020-4394, and R.H.’s baseline funds.

References

- Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2016). TensorFlow: Large-scale machine learning on heterogeneous systems. arXiv:1603.04467 [cs.DC].
- Albert, C., Ulzega, S., Ozdemir, F., Perez-Cruz, F., and Mira, A. (2022). Learning summary statistics for Bayesian inference with autoencoders. arXiv:2201.12059 [cs.LG].
- Ardizzone, L., Kruse, J., Rother, C., and Köthe, U. (2019). Analyzing inverse problems with invertible neural networks. In *Proceedings of the 7th International Conference on Learning Representations (ICLR 2019)*. New Orleans, LA: OpenReview. <https://openreview.net/forum?id=rJed6j0cKX>.
- Baldi, P., Cranmer, K., Faucett, T., Sadowski, P., and Whiteson, D. (2016). Parameterized neural networks for high-energy physics. *The European Physical Journal C*, 76:235.
- Barnes, C., Filippi, S., Stumpf, M., and Thorne, T. (2011). Considerate approaches to achieving sufficiency for ABC model selection. arXiv:1106.6281 [stat.CO].
- Begy, V. and Schikuta, E. (2021). Error-guided likelihood-free MCMC. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, Piscataway, NJ. IEEE.
- Belghazi, M. I., Baratin, A., Rajeshwar, S., Ozair, S., Bengio, Y., Courville, A., and Hjelm, D. (2018). Mutual information neural estimation. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 531–540. PMLR.
- Besag, J. (1986). On the statistical analysis of dirty pictures (with discussion). *Journal of the Royal Statistical Society B*, 48:259–302.
- Bishop, C. (1995). *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, UK.
- Blum, M. G. and Francois, O. (2010). Non-linear regression models for approximate Bayesian computation. *Statistics and Computing*, 20:63–73.
- Blum, M. G., Nunes, M. A., Prangle, D., and Sisson, S. A. (2013). A comparative review of dimension reduction methods in approximate Bayesian computation. *Statistical Science*, 28:189–208.
- Brown, L. D. and Purves, R. (1973). Measurable selections of extrema. *The Annals of Statistics*, 1:902–912.
- Casella, G. and Berger, R. (2001). *Statistical Inference*. Duxbury, Belmont, CA, second edition.
- Chan, J., Perrone, V., Spence, J., Jenkins, P., Mathieson, S., and Song, Y. (2018). A likelihood-free inference framework for population genetic data using exchangeable neural networks. In *Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS 2018)*, pages 8594–8605, Red Hook, NY. Curran.
- Charnock, T., Lavaux, G., and Wandelt, B. D. (2018). Automatic physical inference with information maximizing neural networks. *Physical Review D*, 97:083004.

- Chen, Y., Gutmann, M. U., and Weller, A. (2023). Is learning summary statistics necessary for likelihood-free inference? In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pages 4529–4544. PMLR.
- Chen, Y., Zhang, D., Gutmann, M. U., Courville, A., and Zhu, Z. (2021). Neural approximate sufficient statistics for implicit models. In *Proceedings of the 9th International Conference on Learning Representations (ICLR 2021)*. Virtual: OpenReview. <https://openreview.net/pdf?id=SRDuJssQud>.
- Cobb, A. D., Matejek, B., Elenius, D., Roy, A., and Jha, S. (2023). Direct amortized likelihood ratio estimation. arXiv:2311.10571 [stat.ML].
- Cranmer, K., Brehmer, J., and Louppe, G. (2020). The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117:30055–30062.
- Cranmer, K., Pavez, J., and Louppe, G. (2015). Approximating likelihood ratios with calibrated discriminative classifiers. arXiv:1506.02169 [stat.ML].
- Creel, M. (2017). Neural nets for indirect inference. *Econometrics and Statistics*, 2:36–49.
- Cremer, C., Li, X., and Duvenaud, D. (2018). Inference suboptimality in variational autoencoders. In *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, volume 80, pages 1078–1086. PMLR.
- Cressie, N. (2018). Mission CO₂ntrol: A statistical scientist’s role in remote sensing of atmospheric carbon dioxide. *Journal of the American Statistical Association*, 113:152–168.
- Dai, Z., Damianou, A., González, J., and Lawrence, N. (2015). Variational auto-encoded deep Gaussian processes. arXiv:1511.06455 [cs.LG].
- David, L., Bréon, F.-M., and Chevallier, F. (2021). XCO₂ estimates from the OCO-2 measurements using a neural network approach. *Atmospheric Measurement Techniques*, 14:117–132.
- Davison, A. C., Padoan, S. A., and Ribatet, M. (2012). Statistical modeling of spatial extremes (with discussion). *Statistical Science*, 27:161–201.
- Dayan, P., Hinton, G. E., Neal, R. M., and Zemel, R. S. (1995). The Helmholtz machine. *Neural Computation*, 7:889–904.
- de Castro, P. and Dorigo, T. (2019). INFERNO: Inference-aware neural optimisation. *Computer Physics Communications*, 244:170–179.
- Delaunoy, A., Hermans, J., Rozet, F., Wehenkel, A., and Louppe, G. (2022). Towards reliable simulation-based inference with balanced neural ratio estimation. In *Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS 2022)*, pages 20025–20037, Red Hook, NY. Curran.
- Diggle, P. J. and Gratton, R. J. (1984). Monte Carlo methods of inference for implicit statistical models (with discussion). *Journal of the Royal Statistical Society B*, 46:193–212.
- Dinev, T. and Gutmann, M. U. (2018). Dynamic likelihood-free inference via ratio estimation (DIRE). arXiv:1810.09899 [stat.ML].
- Donsker, M. D. and Varadhan, S. S. (1983). Asymptotic evaluation of certain Markov process expectations for large time. IV. *Communications on Pure and Applied Mathematics*, 36:183–212.

- Drovandi, C. and Frazier, D. T. (2022). A comparison of likelihood-free methods with and without summary statistics. *Statistics and Computing*, 32:42.
- Drovandi, C. C. (2018). ABC and indirect inference. In Sisson, S. A., Fan, Y., and Beaumont, M., editors, *Handbook of Approximate Bayesian Computation*, pages 179–209. Chapman & Hall/CRC Press, Boca Raton, FL.
- Drovandi, C. C., Pettitt, A. N., and Faddy, M. J. (2011). Approximate Bayesian computation using indirect inference. *Journal of the Royal Statistical Society C*, 60:317–337.
- Dyer, J., Cannon, P., Farmer, J. D., and Schmon, S. M. (2024). Black-box Bayesian inference for agent-based models. *Journal of Economic Dynamics and Control*, 161:104827.
- Fan, J. and Yao, Q. (1998). Efficient estimation of conditional variance functions in stochastic regression. *Biometrika*, 85:645–660.
- Fasiolo, M., Pya, N., and Wood, S. N. (2016). A comparison of inferential methods for highly nonlinear state space models in ecology and epidemiology. *Statistical Science*, 31:96–118.
- Fearnhead, P. and Prangle, D. (2012). Constructing summary statistics for approximate Bayesian computation: semi-automatic approximate Bayesian computation (with discussion). *Journal of the Royal Statistical Society B*, 74:419–474.
- Fengler, A., Govindarajan, L., Chen, T., and Frank, M. J. (2021). Likelihood approximation networks (LANs) for fast inference of simulation models in cognitive neuroscience. *eLife*, 10:e65074.
- Ganguly, A., Jain, S., and Watchareeruetai, U. (2023). Amortized variational inference: A systematic review. *Journal of Artificial Intelligence Research*, 78:167–215.
- Gerber, F. and Nychka, D. (2021). Fast covariance parameter estimation of spatial Gaussian process models using neural networks. *Stat*, 10:e382.
- Gershman, S. and Goodman, N. (2014). Amortized inference in probabilistic reasoning. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 36, pages 517–522, Seattle, WA. Cogn. Sci. Soc.
- Gloeckler, M., Deistler, M., Weilbach, C., Wood, F., and Macke, J. H. (2024). All-in-one simulation-based inference. arXiv:2404.09636 [cs.LG].
- Gneiting, T., Balabdaoui, F., and Raftery, A. E. (2007). Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society B*, 69:243–268.
- Gneiting, T. and Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102:359–378.
- Goh, H., Sherifdeen, S., Wittmer, J., and Bui-Thanh, T. (2019). Solving Bayesian inverse problems via variational autoencoders. In *Proceedings of the 2nd Mathematical and Scientific Machine Learning Conference (MSML 2022)*, volume 145, pages 386–425. PMLR.
- Gourieroux, C., Monfort, A., and Renault, E. (1993). Indirect inference. *Journal of Applied Econometrics*, 8:S85–S118.
- Grazian, C. and Fan, Y. (2020). A review of approximate Bayesian computation methods via density estimation: Inference for simulator-models. *Wiley Interdisciplinary Reviews: Computational Statistics*, 12:e1486.
- Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012). A kernel two-sample test. *Journal of Machine Learning Research*, 13:723–773.

- Gutmann, M. U. and Hyvärinen, A. (2012). Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13(11):307–361.
- Hermans, J., Begy, V., and Louppe, G. (2020). Likelihood-free MCMC with amortized approximate ratio estimators. In *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*, volume 119, pages 4239–4248. PMLR.
- Hermans, J., Delaunoy, A., Rozet, F., Wehenkel, A., Begy, V., and Louppe, G. (2022). A crisis in simulation-based inference? Beware, your posterior approximations can be unfaithful. *Transactions on Machine Learning Research*. OpenReview. <https://openreview.net/pdf?id=LHAbHkt6Aq>.
- Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., and Bengio, Y. (2019). Learning deep representations by mutual information estimation and maximization. In *Proceedings of the 7th International Conference on Learning Representations (ICLR 2019)*. New Orleans, LA: OpenReview. <https://openreview.net/forum?id=Bklr3jOcKX>.
- Hoel, D. G. and Mitchell, T. J. (1971). The simulation, fitting, and testing of a stochastic cellular proliferation model. *Biometrics*, 27:191–199.
- Hornik, K. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366.
- Huser, R. and Wadsworth, J. L. (2022). Advances in statistical modeling of spatial extremes. *Wiley Interdisciplinary Reviews: Computational Statistics*, 14:e1537.
- Innes, M. (2018). Flux: Elegant machine learning with Julia. *Journal of Open Source Software*, 3:602.
- Jiang, B., Wu, T.-y., Zheng, C., and Wong, W. H. (2017). Learning summary statistic for approximate Bayesian computation via deep neural network. *Statistica Sinica*, 27:1595–1618.
- Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. (2016). Improving variational autoencoders with inverse autoregressive flow. In *Proceedings of the 30th Conference on Neural Information Processing Systems (NeurIPS 2016)*, pages 4743–4751, Red Hook, NY. Curran.
- Kingma, D. P. and Welling, M. (2013). Auto-encoding variational Bayes. arXiv:1312.6114 [stat.ML].
- Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86.
- Lenzi, A., Bessac, J., Rudi, J., and Stein, M. L. (2023). Neural networks for parameter estimation in intractable models. *Computational Statistics & Data Analysis*, 185:107762.
- Liu, L. and Liu, L. (2019). Amortized variational inference with graph convolutional networks for Gaussian processes. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS 2019)*, volume 89, pages 2291–2300. PMLR.
- Liu, S., Sun, X., Ramadge, P. J., and Adams, R. P. (2020). Task-agnostic amortized inference of Gaussian process hyperparameters. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, pages 21440–21452, Red Hook, NY. Curran.

- Luccioni, A. S., Vignier, S., and Ligozat, A.-L. (2023). Estimating the carbon footprint of BLOOM, a 176B parameter language model. *Journal of Machine Learning Research*, 24:1–15.
- Lueckmann, J.-M., Boelts, J., Greenberg, D., Goncalves, P., and Macke, J. (2021). Benchmarking simulation-based inference. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS 2021)*, volume 130, pages 343–351. PMLR.
- Lueckmann, J.-M., Goncalves, P. J., Bassetto, G., Öcal, K., Nonnenmacher, M., and Macke, J. H. (2017). Flexible statistical inference for mechanistic models of neural dynamics. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS 2017)*, pages 1290–1300, Red Hook, NY. Curran.
- Maceda, E., Hector, E. C., Lenzi, A., and Reich, B. J. (2024). A variational neural Bayes framework for inference on intractable posterior distributions. arXiv:2404.10899 [stat.CO].
- Margossian, C. C. and Blei, D. M. (2023). Amortized variational inference: When and why? arXiv:2307.11018 [stat.ML].
- Mestdagh, M., Verdonck, S., Meers, K., Loossens, T., and Tuerlinckx, F. (2019). Prepaid parameter estimation without likelihoods. *PLoS Computational Biology*, 15:e1007181.
- Miller, B. K., Cole, A., Forré, P., Louppe, G., and Weniger, C. (2021). Truncated marginal neural ratio estimation. In *Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS 2021)*, pages 129–143, Red Hook, NY. Curran.
- Miller, B. K., Cole, A., Weniger, C., Nattino, F., Ku, O., and Grootes, M. W. (2022). swyft: Truncated marginal neural ratio estimation in Python. *Journal of Open Source Software*, 7:4205.
- Mnih, A. and Gregor, K. (2014). Neural variational inference and learning in belief networks. In *Proceedings of the 31st International Conference on Machine Learning (ICML 2014)*, volume 22, pages 1791–1799. PMLR.
- Moores, M. T., Drovandi, C. C., Mengersen, K., and Robert, C. P. (2015). Pre-processing for approximate Bayesian computation in image analysis. *Statistics and Computing*, 25:23–33.
- Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT Press, Cambridge, MA.
- Ong, V. M., Nott, D. J., Tran, M.-N., Sisson, S. A., and Drovandi, C. C. (2018). Variational Bayes with synthetic likelihood. *Statistics and Computing*, 28:971–988.
- Pacchiardi, L. and Dutta, R. (2022a). Likelihood-free inference with generative neural networks via scoring rule minimization. arXiv:2205.15784 [stat.CO].
- Pacchiardi, L. and Dutta, R. (2022b). Score matched neural exponential families for likelihood-free inference. *Journal of Machine Learning Research*, 23:1–71.
- Papamakarios, G. and Murray, I. (2016). Fast ε -free inference of simulation models with Bayesian conditional density estimation. In *Proceedings of the 30th Conference on Neural Information Processing Systems (NeurIPS 2016)*, pages 1036–1044, Red Hook, NY. Curran.
- Papamakarios, G., Sterratt, D., and Murray, I. (2019). Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS 2019)*, volume 89, pages 837–848. PMLR.

- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, pages 7994–8005. Curran, Red Hook, NY.
- Radev, S. T., Mertens, U. K., Voss, A., Ardizzone, L., and Köthe, U. (2022). BayesFlow: Learning complex stochastic models with invertible neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 33:1452–1466.
- Radev, S. T., Schmitt, M., Pratz, V., Picchini, U., Köthe, U., and Buerkner, P.-C. (2023a). JANA: Jointly amortized neural approximation of complex Bayesian models. In *Proceedings of the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023)*, volume 216, pages 1695–1706. PMLR.
- Radev, S. T., Schmitt, M., Schumacher, L., Elsemüller, L., Pratz, V., Schälte, Y., Köthe, U., and Bürkner, P.-C. (2023b). BayesFlow: Amortized Bayesian workflows with neural networks. *Journal of Open Source Software*, 8:5702.
- Rai, S., Hoffman, A., Lahiri, S., Nychka, D. W., Sain, S. R., and Bandyopadhyay, S. (2024). Fast parameter estimation of generalized extreme value distribution using neural networks. *Environmetrics*, 35:e2845.
- Rehn, A. (2022). Amortized Bayesian inference of Gaussian process hyperparameters. Master’s Thesis, University of Helsinki.
- Rezende, D. and Mohamed, S. (2015). Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*, volume 37, pages 1530–1538. PMLR.
- Richards, J., Sainsbury-Dale, M., Zammit-Mangion, A., and Huser, R. (2023). Likelihood-free neural Bayes estimators for censored peaks-over-threshold models. arXiv:2306.15642 [stat.ME].
- Robert, C. P. (2007). *The Bayesian Choice: From Decision-Theoretic Foundations to Computational Implementation*. Springer, New York, NY.
- Ross, G. (1972). Stochastic model fitting by evolutionary operation. In Jeffers, J. N. R., editor, *Mathematical Models in Ecology*, pages 297–308, Oxford, UK. Blackwell Scientific.
- Rozet, F., Delaunoy, A., Miller, B., et al. (2021). LAMPE: Likelihood-free amortized posterior estimation. <https://pypi.org/project/lampe>.
- Rozet, F. and Louppe, G. (2021). Arbitrary marginal neural ratio estimation for simulation-based inference. arXiv:2110.00449 [cs.LG].
- Rudi, J., Bessac, J., and Lenzi, A. (2022). Parameter estimation with dense and convolutional neural networks applied to the FitzHugh–Nagumo ODE. In *Proceedings of the 2nd Mathematical and Scientific Machine Learning Conference (MSML 2022)*, volume 145, pages 781–808. PMLR.
- Sainsbury-Dale, M., Richards, J., Zammit-Mangion, A., and Huser, R. (2023). Neural Bayes estimators for irregular spatial data using graph neural networks. arXiv:2310.02600 [stat.ME].
- Sainsbury-Dale, M., Zammit-Mangion, A., and Huser, R. (2024). Likelihood-free parameter estimation with neural Bayes estimators. *The American Statistician*, 78:1–14.

- Schmitt, M., Bürkner, P.-C., Köthe, U., and Radev, S. T. (2024a). Detecting model misspecification in amortized Bayesian inference with neural networks. In Köthe, U. and Rother, C., editors, *Pattern Recognition. DAGM GCPR 2023*, pages 541–557, Cham, Switzerland. Springer.
- Schmitt, M., Bürkner, P.-C., Köthe, U., and Radev, S. T. (2024b). Detecting model misspecification in amortized Bayesian inference with neural networks: An extended investigation. arXiv:2406.03154 [cs.LG].
- Siahkoohi, A., Rizzuti, G., Orozco, R., and Herrmann, F. J. (2023). Reliable amortized variational inference with physics-based latent distribution correction. *Geophysics*, 88:297–322.
- Sisson, S. A., Fan, Y., and Beaumont, M. (2018). *Handbook of Approximate Bayesian Computation*. Chapman & Hall/CRC Press, Boca Raton, FL.
- Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2021). Score-based generative modeling through stochastic differential equations. In *Proceedings of the 9th International Conference on Learning Representations (ICLR 2021)*. Virtual: OpenReview. <https://openreview.net/forum?id=PxtIG12RRHS>.
- Svendsen, D. H., Hernandez-Lobato, D., Martino, L., Laparra, V., Moreno-Martinez, A., and Camps-Valls, G. (2023). Inference over radiative transfer models using variational and expectation maximization methods. *Machine Learning*, 112:921–937.
- Tejero-Cantero, A., Boelts, J., Deistler, M., Lueckmann, J.-M., Durkan, C., Goncalves, P. J., Greenberg, D. S., and Macke, J. H. (2020). sbi: A toolkit for simulation-based inference. *Journal of Open Source Software*, 5:2505.
- Thomas, O., Dutta, R., Corander, J., Kaski, S., and Gutmann, M. U. (2022). Likelihood-free inference by ratio estimation. *Bayesian Analysis*, 17:1–31.
- Walchessen, J., Lenzi, A., and Kuusela, M. (2023). Neural likelihood surfaces for spatial processes with computationally intensive or intractable likelihoods. arXiv:2305.04634 [stat.ME].
- Wang, Z., Hasenauer, J., and Schälte, Y. (2024). Missing data in amortized simulation-based neural posterior estimation. *PLOS Computational Biology*, 20:e1012184.
- Wiqvist, S., Frellsen, J., and Picchini, U. (2021). Sequential neural posterior and likelihood approximation. arXiv:2102.06522 [stat.ML].
- Wood, S. N. (2010). Statistical inference for noisy nonlinear ecological dynamic systems. *Nature*, 466:1102–1104.
- Zammit-Mangion, A. and Wikle, C. K. (2020). Deep integro-difference equation models for spatio-temporal forecasting. *Spatial Statistics*, 37:100408.
- Zhang, C., Bütepage, J., Kjellström, H., and Mandt, S. (2018). Advances in variational inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41:2008–2026.
- Zhang, L., Ma, X., Wikle, C. K., and Huser, R. (2023). Flexible and efficient spatial extremes emulation via variational autoencoders. arXiv:2307.08079 [stat.ML].
- Åkesson, M., Singh, P., Wrede, F., and Hellander, A. (2022). Convolutional neural networks as summary statistics for approximate Bayesian computation. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 19:3353–3365.

Supplemental Appendix

The appendix is organized as follows. In Section S1 we present a comparative study on a model of spatial extremes. In Section S2 we discuss additional topics. Finally, in Section S3 we provide additional figures and tables relating to the simulation experiment of Section 6.2.

S1 COMPARATIVE STUDY WITH INVERTED MAX-STABLE PROCESSES

Likelihood-based inference with popular models for spatial extremes is notoriously challenging (Huser and Wadsworth, 2022). Among such models, max-stable processes (Padoan et al., 2010, Davison et al., 2012) have perhaps been the most widely used, despite the fact that their likelihood function is computationally intractable in high dimensions (Castruccio et al., 2016). Consider a spatial domain $\mathcal{D} \subset \mathbb{R}^2$. On unit Fréchet margins (i.e., with cumulative distribution function $\exp(-1/z)$, $z > 0$), such models can be represented as $Z(\mathbf{s}) = \sup_{i \geq 1} \xi_i W_i(\mathbf{s})$, $\mathbf{s} \in \mathcal{D}$, where $\{\xi_i\}_{i=1}^\infty$ are points from a Poisson point process with intensity $\xi^{-2} d\xi$ on $(0, \infty)$ and $W_i(\cdot)$ are independent copies of a nonnegative process $W(\cdot)$ with unit mean (also independent of ξ_i). From this representation, one can show that the joint distribution at sites $\{\mathbf{s}_1, \dots, \mathbf{s}_n\} \subset \mathcal{D}$ has the form $\Pr\{Z(\mathbf{s}_1) \leq z_1, \dots, Z(\mathbf{s}_n) \leq z_n\} = \exp\{-V(z_1, \dots, z_n)\}$ with $V(z_1, \dots, z_n) = \mathbb{E}[\max_{j=1, \dots, n} \{W(\mathbf{s}_j)/z_j\}]$ the so-called exponent function. Various choices for the process $W(\cdot)$ lead to different types of max-stable processes: Schlather (2002), for instance, proposed setting $W(\mathbf{s}) = \sqrt{2\pi} \max\{0, \varepsilon(\mathbf{s})\}$, $\mathbf{s} \in \mathcal{D}$, for a standard Gaussian process $\varepsilon(\cdot)$ with some chosen correlation function, which leads to an analytical form for the exponent function, but there are several other possibilities; see Davison et al. (2012), Davison and Huser (2015) and Davison et al. (2019) for exhaustive reviews.

While max-stable processes are often used in applications, recently Huser et al. (2024) argued against the systematic use of max-stable processes in environmental applications due to the rigidity of their dependence structure (especially in the tail) and their inability to capture weakening dependence and asymptotic independence. Huser et al. (2024) instead argued in favour of other more flexible, computationally efficient, and pragmatic alternative modeling solutions. One possible solution to flexibly capture the sub-asymptotic tail behavior of spatial processes displaying asymptotic independence is to consider so-called inverted max-stable processes (Wadsworth and Tawn, 2012). These models effectively swap the roles of the lower and upper tails in max-stable processes: If $Z(\cdot)$ is a max-stable process with unit Fréchet margins, then $\tilde{Z}(\cdot) = 1/Z(\cdot)$ is the corresponding inverted max-stable process, yet now expressed on standard exponential margins.

Unlike max-stable processes, inverted max-stable processes can flexibly capture asymptotic independence in the upper tail, but they remain challenging to fit since they are constructed from max-stable processes themselves, which admit an intractable likelihood function; they are therefore ideal candidates to illustrate the merits of likelihood-free neural approaches summarized in this review paper (as also advocated by Huser et al. (2024)). From their definition, the joint bivariate survival function of the bivariate vector $(\tilde{Z}(\mathbf{s}_j), \tilde{Z}(\mathbf{s}_k))'$, $\mathbf{s}_j, \mathbf{s}_k \in \mathcal{D}$, can be straightforwardly obtained as $\Pr\{\tilde{Z}(\mathbf{s}_j) > z_j, \tilde{Z}(\mathbf{s}_k) > z_k\} = \exp\{-V(1/z_j, 1/z_k)\}$, where $V(\cdot)$ denotes the bivariate restriction of the exponent function of the underlying max-stable process $Z(\cdot)$ to the pair of sites $\{\mathbf{s}_j, \mathbf{s}_k\}$. This formula can be used to derive the bivariate density function $p(Z_j, Z_k \mid \boldsymbol{\theta})$ for any pair of variables $(\tilde{Z}(\mathbf{s}_j), \tilde{Z}(\mathbf{s}_k))'$, $j, k = 1, \dots, n$, where $\boldsymbol{\theta}$ denotes the model parameters. Classical likelihood-based inference then proceeds by maximizing (with respect to $\boldsymbol{\theta}$) a pairwise log-likelihood constructed as $\sum_{j < k} w_{jk} \log\{p(Z_j, Z_k \mid \boldsymbol{\theta})\}$, where $w_{jk} \geq 0$ are suitable weights chosen to improve both statistical and computational

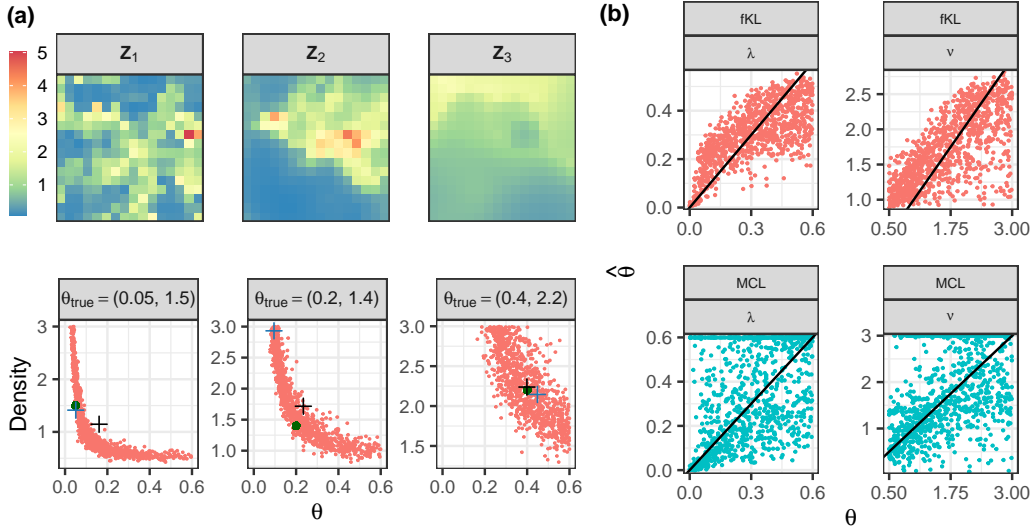


Figure S1: Results from the illustration of Section S1. Panel (a) shows the results from the two methods on three test datasets. The top sub-panels show the data, while the bottom sub-panels show samples from the inferred posterior distribution (red dots), the posterior mean (black cross), the pairwise likelihood estimate (blue cross), as well as the true parameter value (green dot). Panel (b) plots parameter point estimates against the true parameter value in the test dataset for both methods.

efficiency. Often, one sets $w_{jk} = 1$ if $\|\mathbf{s}_j - \mathbf{s}_k\| < \delta_{\max}$ for some cut-off distance $\delta_{\max} > 0$, and $w_{jk} = 0$ otherwise, thus discarding distant pairs of sites. Here, in our simulation study, we consider the Schlather inverted max-stable model parameterized in terms of the length scale $\lambda > 0$ and smoothness parameter $\nu > 0$ of its underlying Matérn correlation function; that is, $\boldsymbol{\theta} = (\lambda, \nu)'$. As for the prior distributions, we let $\lambda \sim \text{Unif}(0, 0.6)$ and $\nu \sim \text{Unif}(0.5, 3)$. We simulate datasets on a 16×16 spatial gridding of $[0, 1]^2$ (i.e., at 256 locations) with a single replicate in each dataset. To make inference on $\boldsymbol{\theta}$, we then consider maximum composite likelihood, MCL, with $\delta_{\max} = 0.2$ (i.e., keeping only about 10% of pairs in the pairwise likelihood for major improvements in speed and accuracy), as well as the fKL minimization approach with a normalizing flow implementation of the approximate posterior distribution using **BayesFlow**.

Figure S1a shows the results when applying the methods to three (test) spatial fields, $\mathbf{Z}_1, \mathbf{Z}_2, \mathbf{Z}_3$. The bivariate posterior distributions are relatively diffuse compared to the single-parameter GP example of Section 6.2, but corroborate the true value in every instance. The MCL estimator also yields reasonable estimates generally, although for the second test case it estimates a value for ν that is on the boundary of the parameter space. Figure S1b plots the posterior means and the MCL estimates for the two parameters. It is well known that estimating these parameters from a single replicate is difficult, but we can see a clear correlation between the posterior mean estimates and the true value; this is less true for the MCL estimates, which often lie on the boundary of the parameter space. The root-mean-squared error between the estimated and the true value by the amortized estimator was about half that of the pairwise likelihood estimator, for both λ (0.119 vs. 0.212) and ν (0.506 vs. 0.842). In this case, these improved estimates (and accompanying full posterior distributions) were also obtained with a 50-fold speedup.

S2 ADDITIONAL TOPICS

This section briefly discusses some important topics related to the principal subject of this review: Section S2.1 gives an overview of neural network architectures commonly used in the methods we reviewed; Section S2.2 describes normalizing flows that are sometimes used to model approximate posterior distributions or likelihood functions; Section S2.3 expands on the approach of Pacchiardi and Dutta (2022) for neural posterior inference through scoring-rule minimization; Section S2.4 discusses conditional generative adversarial networks and variational auto-encoders in the context of amortized inference; Section S2.5 discusses sequential methods and semi-amortization for narrowing the amortization gap within regions of interest in the parameter/sample space; Section S2.6 discusses so-called doubly approximate likelihood approaches; Section S2.7 discusses simulation-based calibration of confidence intervals for neural inference methods; Section S2.8 discusses neural inference under model misspecification; Section S2.9 discusses neural amortization of complex data types; Section S2.10 discusses neural model selection; and finally Section S2.11 gives a non-exhaustive list of applications that have benefitted from amortized inference.

S2.1 Neural Network Architectures

Neural networks are nonlinear functions that are used for function approximation, typically in high-dimensional settings. The functional form of a neural network is known as its architecture. Throughout the review we generally treat the neural network as a black-box function characterized by parameters that need to be optimized according to some criterion. The purpose of this section is to give a brief insight into the black boxes that are most pertinent to this field. Since neural networks are used in various ways when making amortized inference, here we keep the notation general and denote the input to the neural network as \mathbf{X} and the output as \mathbf{Y} , which we assume are real-valued and of different dimension. The type of neural network used to capture a specific input/output relationship largely depends on the multivariate structure of the input \mathbf{X} .

Unstructured data: When the input \mathbf{X} has no discernible multivariate structure (i.e., it is not spatially, temporally, or otherwise structured), a commonly-used architecture is the multi-layer perceptron (MLP, e.g., Bishop, 1995, Chapter 5). For some $L > 1$ layers, the MLP is a hierarchy of so-called fully-connected layers,

$$\begin{aligned}\mathbf{Y} &= \varphi_L(\mathbf{W}_L \mathbf{h}_{L-1} + \mathbf{b}_L), \\ \mathbf{h}_l &= \varphi_l(\mathbf{W}_l \mathbf{h}_{l-1} + \mathbf{b}_l), \quad l = 2, \dots, L-1, \\ \mathbf{h}_1 &= \varphi_1(\mathbf{W}_1 \mathbf{X} + \mathbf{b}_1),\end{aligned}\tag{S1}$$

where, for $l = 1, \dots, L$, $\boldsymbol{\gamma}_l \equiv (\text{vec}(\mathbf{W}_l)', \mathbf{b}_l')'$ are the neural-network parameters at layer l comprising both the matrix of weights \mathbf{W}_l and vector of biases \mathbf{b}_l of appropriate dimension, and $\varphi_l(\cdot)$ are non-linear activation functions applied elementwise over their arguments. Popular activation functions include the sigmoid function, the hyperbolic tangent function, and the rectified linear unit, which returns its input multiplied by the Heaviside step function. The quantities $\mathbf{h}_l, l = 1, \dots, L-1$, are often referred to as hidden states. Training the network is the process of estimating the neural-network parameters $\boldsymbol{\gamma} \equiv (\boldsymbol{\gamma}_1', \dots, \boldsymbol{\gamma}_L')'$ by minimizing some empirical risk function, usually via stochastic gradient descent. The MLP is highly parameterized, and is therefore most often used as a small component of the more parsimonious architectures discussed below.

Gridded data: The mainstream neural network architecture used with gridded data is the convolutional neural network (CNN, LeCun et al., 1998). Assume, for ease of exposition, that $\mathbf{X} \in \mathbb{R}^n$ are ordered pixel values of a 1-D image, where n is the number of pixels. Despite their name, CNNs typically implement cross-correlations and not convolutions:

an output of the first layer of a 1-D CNN is

$$h_{1,i} = \varphi_1 \left(\sum_{j=1}^{n_K} \tilde{X}_{i+j-1} K_{1,j} + b_{1,i} \right); \quad i = 1, \dots, n,$$

where $\mathbf{h}_1 \equiv (h_{1,1}, \dots, h_{1,n})'$, $\mathbf{K}_1 \equiv (K_{1,1}, \dots, K_{1,n_K})'$ is an n_K -dimensional discrete kernel, and $\tilde{\mathbf{X}}$ is an appropriately padded extension of \mathbf{X} of size $n + n_K - 1$, where typically the padded values are equal to zero. Subsequent convolutional layers take the same form and training then involves estimating $\gamma_l \equiv (\mathbf{K}'_l, \mathbf{b}'_l)'$ for $l = 1, \dots, L$. The kernels play the role of weight sharing, and thus CNN layers are considerably more parsimonious than fully-connected ones. In practice, inputs are often 2-D, and many “convolutions” are done in each layer, so that a set of convolved quantities known as channels, are input to the subsequent layer. When one has multiple 2-D image channels input to a layer, 3-D convolutional filters are used to do the convolution. The final layer of a CNN is typically an MLP that maps the output of the convolutional layers to the output. See, for example, Gerber and Nychka (2021), Lenzi et al. (2023), or Richards et al. (2023) for examples of CNN-based neural Bayes estimator used to make amortized inference with gridded spatial data.

Time series/spatio-temporal data: Time series data are also highly structured, and the 1-D CNN described above is often used for time series data (e.g., Dinev and Gutmann, 2018, Åkesson et al., 2022, Rudi et al., 2022); extensions have been proposed for using them for estimating parameters in a spatio-temporal setting (e.g., de Bézenac et al., 2018, Zammit-Mangion and Wikle, 2020). However, CNNs do not take into account the temporal ordering of the data, and do not explicitly account for temporal dynamics. A classic architecture that takes temporal ordering into account is the recurrent neural network (RNN, Williams and Zipser, 1989). A layer in a vanilla RNN is given by

$$\mathbf{h}_l = \varphi_l(\mathbf{W}\mathbf{h}_{l-1} + \mathbf{b} + \mathbf{U}\mathbf{x}_{l-1}),$$

where l now denotes the position in a sequence (i.e., the time point), \mathbf{x}_{l-1} is the data (e.g., a covariate) at $l - 1$, and \mathbf{U} is a matrix that maps the input to the hidden states. Since the matrices \mathbf{W} and \mathbf{U} , and the bias \mathbf{b} , are time invariant, the RNN is reasonably parsimonious. A popular variant of the RNN is the long-short-term-memory (LSTM, Hochreiter and Schmidhuber, 1997) model, which is designed to capture longer-range dependencies. Despite successes in various machine learning applications, both RNNs and LSTMs are being superseded by transformer-based architectures that incorporate so-called self-attention modules intended to capture long-range dependencies and that are amenable to more efficient training algorithms (Vaswani et al., 2017).

Graphical data and irregular spatial data: Graph neural networks (GNNs) are designed for use with data that can be represented as a graph, where entities (nodes) and their relationships (edges) encapsulate the data structure (e.g., Zhang et al., 2019, Zhou et al., 2020, Wu et al., 2021). When performing “graph-level regression”, where the entire graph (e.g., data) is associated with some fixed-dimensional vector (e.g., unknown model parameters) that we wish to infer, a GNN typically consists of three modules: a propagation module, a readout module, and a mapping module. The propagation module consists of several layers of graph operations. For layer l , node j , a possible GNN propagation layer is given by

$$\mathbf{h}_{l,j} = \varphi_l \left(\mathbf{W}_{l,1} \mathbf{h}_{l-1,j} + \frac{1}{|\mathcal{N}(j)|} \sum_{j' \in \mathcal{N}(j)} a_{\beta_l}(j, j') \mathbf{W}_{l,2} \mathbf{h}_{l-1,j'} + \mathbf{b}_l \right),$$

where $\mathbf{h}_{l,j}$ is the hidden state of the j th node at layer l , $\mathcal{N}(j)$ denotes the set of neighbors of node j , and $a_{\beta_l}(j, j')$ is a weight that is a function of properties associated with nodes j

and j' (e.g., the spatial distance between the nodes), parameterized by β_l . The trainable parameters at layer l are therefore given by $\gamma_l \equiv (\text{vec}(\mathbf{W}_{l,1})', \text{vec}(\mathbf{W}_{l,2})', \mathbf{b}_l', \beta_l')'$. The readout module maps the hidden state at the final layer of the propagation module to a vector of fixed dimension by summarizing each dimension separately (e.g., by averaging), while the mapping module maps this fixed length vector to the output \mathbf{Y} , typically using an MLP. See Sainsbury-Dale et al. (2023) for an example of GNN-based neural Bayes estimator used to make amortized inference with spatial data observed at irregular locations.

Exchangeable data: Inferences made from exchangeable data (e.g., data with independent replicates) should be invariant to permutations of the data. This permutation-invariance property is satisfied by design with the neural network architecture commonly known as DeepSets (Zaheer et al., 2017). Consider an input consisting of m replicates, $\mathbf{X} \equiv (\mathbf{X}'_1, \dots, \mathbf{X}'_m)'$. The DeepSets architecture is:

$$\mathbf{Y} = \phi(\mathbf{a}[\{\psi(\mathbf{X}_i) : i = 1, \dots, m\}]), \quad (\text{S2})$$

where $\phi(\cdot)$ is typically an MLP, $\mathbf{a}(\cdot)$ is a permutation-invariant set function which combines its set elements through elementwise operations (e.g., elementwise addition or average), and $\psi(\cdot)$ is a neural network whose structure depends on that of each input datum (e.g., $\psi(\cdot)$ could be a CNN if \mathbf{X}_i is gridded spatial data). Note that $\psi(\cdot)$ is common to all replicates in the dataset; any function of the form of Equation S2 is guaranteed to be permutation invariant. See Chan et al. (2018) for a DeepSets-based approximate posterior distribution for making inference with population-genetics data, and Sainsbury-Dale et al. (2024) for a DeepSets-based neural Bayes estimator used to make amortized inference with replicated spatial data.

S2.2 Normalizing Flows

Approximate posterior distributions or likelihood functions in amortized inference are often modeled using normalizing flows, which map a target density to a simple, reference, density. Consider an approximate posterior distribution for illustration. Let $\boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^d$ be the target variable of interest whose density we wish to model, and let $\tilde{\boldsymbol{\theta}} \in \tilde{\Theta} \subseteq \mathbb{R}^d$ denote a reference variable whose density is known; here we take this to be a spherical Gaussian density. A normalizing flow is a diffeomorphism $\mathbf{g} : \Theta \rightarrow \tilde{\Theta}$ parameterized by some parameters $\boldsymbol{\kappa}$ such that $\mathbf{g}(\boldsymbol{\theta}; \boldsymbol{\kappa}) = \tilde{\boldsymbol{\theta}} \sim \text{Gau}(\mathbf{0}, \mathbf{I})$. By the change-of-variables formula, the target log-density is

$$\log q(\boldsymbol{\theta}; \boldsymbol{\kappa}) = -\frac{d}{2} \log(2\pi) - \frac{1}{2} \|\mathbf{g}(\boldsymbol{\theta}; \boldsymbol{\kappa})\|^2 + \log |\det \nabla \mathbf{g}(\boldsymbol{\theta}; \boldsymbol{\kappa})|. \quad (\text{S3})$$

Normalizing flows shift the burden from modeling a density over a high-dimensional space to that of modeling a flexible differentiable invertible mapping $\mathbf{g}(\cdot; \boldsymbol{\kappa})$. Recall from Section 3 that amortization can be imbued into the learning problem by making the parameters $\boldsymbol{\kappa}$ themselves outputs of neural networks, which we denote as $\boldsymbol{\kappa}_\gamma(\mathbf{Z})$.

There are several types of normalizing flows and excellent reviews include those by Kobzyev et al. (2020) and Papamakarios et al. (2021). Here, we outline two popular ones used in an amortized inferential setting:

Affine coupling flows: An affine coupling flow is a composition of several affine coupling blocks constructed as follows. Consider the l th coupling block in a flow comprising L such blocks, that takes in $\tilde{\boldsymbol{\theta}}^{(l-1)}$ as input, and that outputs $\tilde{\boldsymbol{\theta}}^{(l)}$, with $\tilde{\boldsymbol{\theta}}^{(0)} \equiv \boldsymbol{\theta}$ and $\tilde{\boldsymbol{\theta}}^{(L)} \equiv \tilde{\boldsymbol{\theta}}$. Each affine coupling block partitions the input $\tilde{\boldsymbol{\theta}}^{(l-1)}$ into two disjoint components, $\{\tilde{\boldsymbol{\theta}}_1^{(l-1)}, \tilde{\boldsymbol{\theta}}_2^{(l-1)}\}$, and then outputs

$$\begin{aligned} \tilde{\boldsymbol{\theta}}_1^{(l)} &= \tilde{\boldsymbol{\theta}}_1^{(l-1)}, \\ \tilde{\boldsymbol{\theta}}_2^{(l)} &= \tilde{\boldsymbol{\theta}}_2^{(l-1)} \odot \exp\{\boldsymbol{\kappa}_{\gamma,1}^{(l)}(\tilde{\boldsymbol{\theta}}_1^{(l-1)}, \mathbf{Z})\} + \boldsymbol{\kappa}_{\gamma,2}^{(l)}(\tilde{\boldsymbol{\theta}}_1^{(l-1)}, \mathbf{Z}), \end{aligned}$$

where $\kappa_{\gamma}(\cdot) \equiv \{\kappa_{\gamma}^{(1)}(\cdot), \dots, \kappa_{\gamma}^{(L)}(\cdot)\}$ with $\kappa_{\gamma}^{(l)} \equiv \{\kappa_{\gamma,1}^{(l)}(\cdot), \kappa_{\gamma,2}^{(l)}(\cdot)\}$, $l = 1, \dots, L$, are (generic, not necessarily invertible) neural networks that take in both the data and subsets of the input/output variables. The mapping of each block is invertible, with inverse given by,

$$\begin{aligned}\tilde{\theta}_1^{(l-1)} &= \tilde{\theta}_1^{(l)}, \\ \tilde{\theta}_2^{(l-1)} &= \{\tilde{\theta}_2^{(l)} - \kappa_{\gamma,2}^{(l)}(\tilde{\theta}_1^{(l)}, \mathbf{Z})\} \odot \exp\{-\kappa_{\gamma,1}^{(l)}(\tilde{\theta}_1^{(l)}, \mathbf{Z})\}.\end{aligned}$$

The Jacobian of the transformation is triangular and therefore its determinant is straightforward to compute. Note that the variables need to be permuted between coupling blocks in order for all input components to (eventually) be transformed. Affine coupling flows find their roots in the work of Dinh et al. (2016), and one of its variants is implemented in the popular package **BayesFlow** that does amortized inference by forward KL minimization (Section 3.2.1).

Masked autoregressive flows: A popular variant of the autoregressive flow is a composition of several layers of the form,

$$\begin{aligned}\tilde{\theta}_1^{(l)} &= h\{\tilde{\theta}_1^{(l-1)}; \kappa_{\gamma,1}^{(l)}(\mathbf{Z})\}, \\ \tilde{\theta}_t^{(l)} &= h\{\tilde{\theta}_t^{(l-1)}; \kappa_{\gamma,t}^{(l)}(\tilde{\theta}_{1:(t-1)}^{(l-1)}, \mathbf{Z})\}, \quad t = 2, \dots, d,\end{aligned}$$

where $\kappa^{(l)}(\cdot) \equiv \{\kappa_{\gamma,1}^{(l)}(\cdot), \kappa_{\gamma,2}^{(l)}(\cdot), \dots, \kappa_{\gamma,d}^{(l)}(\cdot)\}$ are (generic, not necessarily invertible) neural networks that output the parameters of an invertible transformation $h(\cdot)$. The resulting Jacobian is triangular and hence the determinant required for the transformation is easy to compute. There are several connections between the vanilla autoregressive flow presented here and the affine coupling flows; these are discussed in detail by Papamakarios et al. (2021).

A computationally attractive implementation of the autoregressive flow is the masked autoregressive flow, where $\kappa^{(l)}(\cdot)$ is a single neural network that outputs the required parameters for all $t = 1, \dots, d$. This is achieved by ensuring there are no connections between $\tilde{\theta}_{t:d}^{(l-1)}$ and the flow parameters corresponding to the t th transformation, through appropriate use of masking 0-1 matrices. Masked autoregressive flows were developed by Papamakarios et al. (2017), and have been extensively used for both amortized posterior inference and amortized likelihood estimation (e.g., Greenberg et al., 2019, Papamakarios et al., 2019, Brehmer et al., 2020, Wqvist et al., 2021, Glöckler et al., 2022).

S2.3 Neural posterior inference via scoring rule minimization

Pacchiardi and Dutta (2022) implicitly define an approximate posterior distribution through a (generally non-invertible) conditional transformation of a random variable with a simple known distribution. Specifically, they let \mathbf{W} be a latent variable with known distribution and construct a (generally non-invertible) neural network $\mathbf{f}_{\gamma}(\mathbf{Z}, \mathbf{W})$, with parameters γ , that takes both \mathbf{W} and the data \mathbf{Z} as input, and outputs θ . The variable \mathbf{W} establishes an implicit conditional distribution of θ for a given \mathbf{Z} , which we denote by $q_{\gamma}(\theta; \mathbf{Z})$. Then, instead of minimizing the forward KL divergence as in Equation 7, they minimize an expected score, that is, they solve

$$\gamma^* = \arg \min_{\gamma} \sum_{i=1}^N s\{q_{\gamma}(\cdot; \mathbf{Z}^{(i)}), \theta^{(i)}\}, \quad (\text{S4})$$

where $\theta^{(i)} \sim p(\theta)$, $\mathbf{Z}^{(i)} \sim p(\mathbf{Z} \mid \theta^{(i)})$, and $s(\cdot, \cdot)$ is a proper scoring rule (Gneiting et al., 2007), such as the energy score. The score is intractable (since $q_{\gamma}(\cdot; \mathbf{Z})$ cannot be evaluated), but often one can estimate it for a given $\theta^{(i)}$ and $\mathbf{Z}^{(i)}$ with a Monte Carlo approximation using samples from $q_{\gamma}(\cdot; \mathbf{Z}^{(i)})$, which are straightforward to generate by sampling

\mathbf{W} from its known distribution and computing $\mathbf{f}_\gamma(\mathbf{Z}^{(i)}, \mathbf{W})$ for each realization of \mathbf{W} . The objective function in Equation S4 is a generalization of that in Equation 7, and the two are identical if $s(\cdot, \cdot)$ is chosen to be the logarithmic score. The approach of Pacchiardi and Dutta (2022) can thus be seen as a form of amortized approximate generalized Bayesian inference (Bissiri et al., 2016).

S2.4 Likelihood-free reverse-KL approaches

In this section we outline two additional methods related to reverse-KL minimization that are likelihood-free by design: conditional generative adversarial networks, and variational auto-encoders.

Conditional generative adversarial networks: Generative adversarial networks (GANs Goodfellow et al., 2014) are widely known for their ability to generate realistic high-dimensional data, such as text, images, and video. They comprise two neural networks, a generator, which generates outputs of interest from some latent noise input that is arbitrarily distributed (e.g., normally distributed), and a discriminator whose role is to discriminate outputs from the generator and data from the underlying data generating process. GANs are trained using a minimax loss function, where the discriminator parameters are optimized to be able to correctly classify a datum as coming from the data generating process or from the generator, and the generator network is optimized to generate data that resemble samples from the true generating process as much as possible.

When used in the context of simulation-based inference, the generator in a GAN generates samples from a posterior distribution through the use of data \mathbf{Z} as a conditioning input (Ramesh et al., 2022). Specifically, the inference network in the conditional GAN is the generator $\mathbf{g}_\gamma(\mathbf{Z}, \mathbf{W})$, that takes in the conditioning data \mathbf{Z} and latent quantities \mathbf{W} as input (that are independent from \mathbf{Z} , arbitrarily distributed, and that can be easily sampled), and returns parameters $\boldsymbol{\theta}$. For any fixed \mathbf{Z} it can be shown that, once trained, $\boldsymbol{\theta} \sim q_\gamma(\boldsymbol{\theta}; \mathbf{Z})$, where $q_\gamma(\cdot; \mathbf{Z})$ minimizes the reverse KL divergence between the true posterior distribution and the approximate posterior distribution (Ramesh et al., 2022, Appendix A.2). Specifically, generator parameters γ are the solution to

$$\gamma^* = \arg \min_{\gamma} \mathbb{E}_{\mathbf{Z}} [\text{KL}\{q_\gamma(\boldsymbol{\theta}; \mathbf{Z}) \parallel p(\boldsymbol{\theta} \mid \mathbf{Z})\}].$$

Training of the GANs is sample-based and, although it yields a variational approximate posterior distribution, it does not require knowledge of the likelihood function.

Variational auto-encoders: The variational auto-encoder (VAE, Kingma and Welling, 2013) is a type of amortized variational inference where the likelihood is assumed to have a simple form with distributional parameters expressed through a decoder (the inference network is analogously referred to as an encoder). Consider the case where the assumed likelihood function $q\{\mathbf{Z}; \boldsymbol{\mu}_\eta(\boldsymbol{\theta}), \boldsymbol{\Sigma}_\eta(\boldsymbol{\theta})\}$ is a Gaussian function, with mean parameter $\boldsymbol{\mu}_\eta(\boldsymbol{\theta})$ and covariance matrix $\boldsymbol{\Sigma}_\eta(\boldsymbol{\theta})$ outputs of a neural network that has parameters $\boldsymbol{\eta}$. Then, given a sample $\boldsymbol{\theta}^{(j)}$ from the variational distribution over $\boldsymbol{\theta}$, one produces a variational posterior-predictive sample from $q\{\mathbf{Z}; \boldsymbol{\mu}_\eta(\boldsymbol{\theta}^{(j)}), \boldsymbol{\Sigma}_\eta(\boldsymbol{\theta}^{(j)})\}$. In a VAE, the likelihood network and the inference network are trained jointly, and the optimization problem of Equation 8 in the main text becomes

$$(\gamma^{*'}, \boldsymbol{\eta}^{*'})' \approx \arg \min_{(\gamma', \boldsymbol{\eta}')'} \sum_{k=1}^K \sum_{j=1}^J \left(\log q\{\boldsymbol{\theta}^{(j)}; \boldsymbol{\kappa}_\gamma(\mathbf{Z}^{(k)})\} - \log[q\{\mathbf{Z}^{(k)}; \boldsymbol{\mu}_\eta(\boldsymbol{\theta}^{(j)}), \boldsymbol{\Sigma}_\eta(\boldsymbol{\theta}^{(j)})\} p(\boldsymbol{\theta}^{(j)})] \right),$$

where $\mathbf{Z}^{(k)} \sim p(\mathbf{Z})$ and $\boldsymbol{\theta}^{(j)} \sim q\{\boldsymbol{\theta}; \boldsymbol{\kappa}_\gamma(\mathbf{Z}^{(k)})\}$. Although the VAE has been used in an amortized inference framework by Goh et al. (2019), it is most often used in situations

where one only has realizations of \mathbf{Z} and no knowledge of the underlying data generating process. In these cases $\boldsymbol{\theta}$ plays the role of a latent variable encoding a lower dimensional representation of the data (e.g., Doersch, 2016, Kingma et al., 2019, Cartwright et al., 2023).

S2.5 Mitigating the amortization gap with sequential methods and semi-amortization

The amortization gap, which all methods discussed in this review may suffer from, can be reduced by adopting a sequential strategy, where re-training of neural networks is done in regions of interest. Sequential methods are not-amortized *per se* (since the region of interest varies with the dataset) but can build and extend on the methods discussed in this review. In sequential methods for posterior inference, parameters are sampled from a proposal distribution that generally differs from the prior distribution, and require importance-weight adjustment to the loss function (e.g., Lueckmann et al., 2017) or to the approximate posterior distribution post-training (e.g., Papamakarios and Murray, 2016); see also Greenberg et al. (2019) and Goncalves et al. (2020). Sequential methods are also used to refine the neural likelihood function in region of high parameter interest (Papamakarios et al., 2019) and can also be used to specifically lower the amortization gap globally. For example, using a Bayesian neural network for the likelihood network, Lueckmann et al. (2019) propose a training strategy that sequentially minimizes the uncertainty on the likelihood approximation. Another approach to deal with the amortization gap when making posterior inference is semi-amortization. Here, a global amortized inference network is used as an initialization for standard (non-amortized) inference. This is attractive in the reverse-KL scenario, where (non-amortized) variational inference is straightforward; see, for example, Hjelm et al. (2016) and Kim et al. (2018).

S2.6 Neural “doubly-approximate” likelihood approaches

Some amortized neural likelihood methods proposed in the recent literature are doubly-approximate in that two levels of approximation are involved: (i) neural networks are used to approximate a certain target and (ii) this target is itself an approximate likelihood (rather than the true likelihood). Here we outline two such methods.

Kernel density target: Fengler et al. (2021) propose using so-called likelihood approximation networks (LANs), which are divided into two distinct methods: the “pointwise approach” and the “histogram approach”. The pointwise approach learns the log-likelihood function $\log p(\mathbf{Z} | \boldsymbol{\theta})$ by representing it through a neural network taking a parameter vector and a data point as (multi-dimensional) input and returning the log-likelihood value as (uni-dimensional) output. To train the network, empirical likelihood values are first obtained for each specific parameter value $\boldsymbol{\theta}$ using a kernel density estimator of the form $\hat{p}_h(\mathbf{Z} | \boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^N f_h(\|\mathbf{Z} - \mathbf{Z}^{(i)}\|)$ for some kernel function $f_h(\cdot)$ with bandwidth $h > 0$ (e.g., a centred Gaussian density with standard deviation h), where $\mathbf{Z}^{(i)} \sim p(\mathbf{Z} | \boldsymbol{\theta})$; these empirical values serve as surrogates for the (unknown) true likelihood values. Then, the neural approximate likelihood can be trained by minimizing an appropriate loss function contrasting neural log-likelihood and empirical log-likelihood values $\hat{p}_h(\mathbf{Z} | \boldsymbol{\theta})$. Under the squared (respectively absolute) error loss, the pointwise LAN approach boils down to performing a deep non-linear mean (respectively median) regression with empirical likelihood values playing the role of response variables and the combination of parameters and data points playing the role of covariates. Fengler et al. (2021) instead consider the Huber loss (Huber, 1964) due to its robustness, but many other options are also possible. When the dataset contains multiple independent replicates, the overall approximate log-likelihood function is then obtained by evaluating, one by one, all single-observation log-likelihoods using the trained network and then summing them up. This pointwise approach admits graphical representation shown in Figure S2.

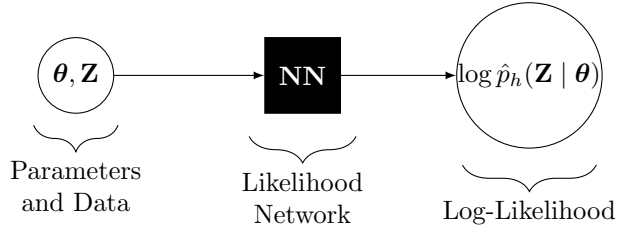


Figure S2: Graphical representation of a likelihood network that takes both data and parameters as input and that outputs an estimate of the log likelihood.

The histogram approach, in contrast, seeks to provide a method that can directly evaluate the overall (log-)likelihood function for an arbitrary number of data points via a single forward pass through the network. This is achieved by modifying the response variable to be binned empirical likelihood values (using the same kernel density estimator technique), and regressing it against parameter values alone using an appropriate neural network architecture and loss function (e.g., the KL divergence between binned densities).

While both of these approaches were shown by Fengler et al. (2021) to provide good results in a relatively low-dimensional cognitive neuroscience data example (using a multi-layer perceptron for the neural network in the pointwise approach, and a different network with convolutional layers in the histogram approach), LANs have several drawbacks (Boelts et al., 2022). First, the response variables in the described deep regression problem are not true likelihood values, but based on empirical kernel-density likelihood values, which are approximate and have their own limitations, especially in their ability to capture the tails. Second, since the accuracy of the method depends on the accuracy of the kernel-density-estimated empirical likelihoods, a large number of training samples are typically required to ensure that the empirical kernel-density likelihood values accurately approximate their true likelihood counterparts. For example, Fengler et al. (2021) used 1.5 million parameter values and 100,000 data points per parameter, i.e., more than 100 billion training data points in total. This requirement is likely to make the training phase computationally prohibitive in many applications.

Vecchia likelihood target: Another recently-introduced doubly-approximate approach consists in exploiting the Vecchia likelihood approximation (Vecchia, 1988, Stein et al., 2004, Katzfuss and Guinness, 2021), which represents the likelihood function as a product of univariate conditional densities, yet with a reduced number of conditioning variables (hence the approximation); then, amortized neural conditional density estimators can be trained for each density of the form $p(Z_j | \mathbf{Z}_{\text{Nei}(j)})$, where Z_j denotes the j th element of the vector \mathbf{Z} and $\mathbf{Z}_{\text{Nei}(j)}$ contains some neighboring variables within its “history” according a predefined sequence; after training, these approximate conditional densities are put back together to get an amortized likelihood approximation of $p(\mathbf{Z} | \boldsymbol{\theta})$ (Majumder and Reich, 2023, Majumder et al., 2023).

S2.7 Reliable inference with simulation-based calibration

Due to the amortization gap, credible and confidence intervals/regions obtained from the neural inference approaches described in this review paper can be over-confident with poor frequentist properties, thus biasing inferences and subsequent decision-making (Hermans et al., 2022). Various methods have been proposed to mitigate this issue. One possible approach called WALDO (Masserano et al., 2023), which builds on the work of Dalmaso et al. (2021), aims at constructing confidence regions with finite-sample conditional validity through Neyman inversion, mimicking the well-known Wald test. This method considers a Wald-like test statistic of the form $\mathbf{W}(\mathbf{Z}; \boldsymbol{\theta}_0) = \{\mathbb{E}(\boldsymbol{\theta} | \mathbf{Z}) - \boldsymbol{\theta}_0\}' \mathbb{V}(\boldsymbol{\theta} | \mathbf{Z})^{-1} \{\mathbb{E}(\boldsymbol{\theta} | \mathbf{Z}) - \boldsymbol{\theta}_0\}$, where $\mathbb{E}(\boldsymbol{\theta} | \mathbf{Z})$ and $\mathbb{V}(\boldsymbol{\theta} | \mathbf{Z})$ are, respectively, the conditional mean

and the conditional covariance matrix of the model parameter θ given data \mathbf{Z} , which can be estimated from the neural amortized method under consideration, either analytically or by sampling. The critical values of this test statistic can be learned through a separate deep quantile regression model. Confidence regions resulting from WALDO will, by construction, satisfy conditional coverage regardless of the sample size and the true value of θ , provided the quantile regression fits well. WALDO is applicable to both frequentist and Bayesian neural methods, and when used in a Bayesian context it still yields valid confidence regions at the desired level irrespective of the prior distribution. Other empirical simulation-based approaches for validating an amortized estimator include those of Talts et al. (2018), Zhao et al. (2021), and Hermans et al. (2022).

S2.8 Inference under model misspecification

Model misspecification (i.e., distribution shift) is a problem that is ubiquitous in statistical modeling and inference. With likelihood-based inference, the effect of model misspecification is well understood from classical asymptotic theory (see, e.g., Davison, 2003, page 147–148); the same is not true with neural inference approaches. This open research question is especially relevant as neural networks are known to extrapolate poorly in general. While there are simple numerical experiments that suggest that these methods have some built-in robustness against misspecification (e.g., supplementary material of Sainsbury-Dale et al. (2024)), others suggest otherwise (e.g., Bon et al., 2023). Efforts to tackle model misspecification include the use of distribution-mismatch-detection of the learned summary statistics with respect to a reference distribution (Radev et al., 2023, Schmitt et al., 2024). Kelly et al. (2023) instead modify summary statistics using auxiliary parameters to build a robust version of the sequential neural likelihood approach of Papamakarios et al. (2019).

S2.9 Inference with complex data types

Certain neural inference approaches, for example those relying on normalizing flows (e.g., Papamakarios et al., 2019), cannot be easily applied when the data are partially discrete and partially continuous. To address this issue, Boelts et al. (2022) propose the so-called mixed neural likelihood estimation (MNLE) method, which draws on Papamakarios et al. (2019), but splits the inference problem into two simpler sub-problems: one that trains a conditional density network (e.g., using normalizing flows) to model continuous data given model parameters and the discrete data (used as an input to the neural network), and another one that only models discrete data given model parameters. These two networks are then combined together to get the full model.

Similarly, neural inference with missing or censored data is not trivial because most neural network architectures are not made to handle such inputs. Graph neural networks (GNNs, see Section S2.1), designed to take different graphical structures as input, is a possible architecture that can naturally deal with data missingness. Alternatively, Wang et al. (2024) propose setting the missing data to an arbitrary constant, and augmenting the input with an additional binary dataset of the same size indicating which data are missing. A neural network can then be trained on this modified and augmented input; the rationale is that the network will learn to treat the missing data differently from the non-missing observations. The drawback of this approach, however, is that the trained neural network is sensitive to the data missingness model used during the training phase, which can bias inference if misspecified. In the context of censored data, Richards et al. (2023) use an approach similar to Wang et al. (2024) for making inference in a spatial extremes application where non-extreme values are censored (but not missing). There is, however, an important difference with respect to Wang et al. (2024), namely that the censoring mechanism is fully determined by the data and the censoring threshold, and thus never misspecified.

S2.10 Neural model selection

Amortized model comparison and selection can naturally proceed by training a neural discriminator taking data (or summary statistics thereof) as input and returning a model probability as output, thus mimicking the architecture of neural Bayes estimators (recall Section 3.1) but with a modified final output layer. Such an approach is closely related to hypothesis testing, and it has been used, for example, by Ahmed et al. (2022) to select among models with different tail structures. Alternatively, another possible model selection approach is to proceed by computing Bayes factors (i.e., ratios of marginal likelihoods), which can be directly obtained when the posterior density and the likelihood function are both available, as in the JANA framework (Radev et al., 2023). Traditional likelihood-ratio tests to compare nested models can also be performed using (approximate) neural full likelihood and likelihood-ratio methods, even when they are only available up to a normalizing constant.

S2.11 Applications employing amortized neural inference

There has been a marked increase in the use of neural networks for making parameter inference in recent years. The following list of applications is non-exhaustive, but serves to show the wide scope of applications neural amortized inference can be useful for.

David et al. (2021) use a neural point estimator to map spectra collected by NASA’s OCO-2 satellite to column-averaged carbon dioxide, using estimates available from classical inversion techniques as target output parameters during training. Amortization in this application is highly desirable because of the high throughput of remote sensing instruments. Richards et al. (2023) apply neural Bayes estimators to fit hundreds of thousands of models for spatial extremes (which have computationally-intractable likelihoods) to censored particulate matter data in Saudi Arabia in a study on air pollution.

In the context of approximate Bayes, Speiser et al. (2017) use amortized variational methods to infer neural activity from fluorescence traces; their inference network uses a combination of recurrent neural networks and convolutional neural networks (see Section S2.1), and as approximate posterior distribution they use a Bernoulli distribution for the spike probabilities. Chan et al. (2018) use forward KL minimization with a Gaussian approximate posterior distribution of the intensity of recombination hotspots from genome data. Siahkoohi et al. (2023) solve a geophysical inverse problem to infer the state of Earth’s subsurface from surface measurements. They use forward KL minimization with hierarchical normalizing flows (Kruse et al., 2021) to approximate the posterior distribution of the state. Similarly, von Krause et al. (2022) use forward KL minimization to fit diffusion models to response time data from over one million participants and study how mental speed varies with age.

There have also been many applications of neural likelihood and likelihood-to-evidence-ratio approximators. For instance, Lueckmann et al. (2019) construct neural synthetic likelihoods to make inference on the evolution of membrane potential in (brain) neurons through the Hodgkin–Huxley biophysical model. Fengler et al. (2021) apply the LAN method to make inference with stochastic differential equation models commonly used in cognitive neurosciences. Using amortized neural likelihood ratio approximators, Baldi et al. (2016) make inference with high-energy particle physics models; Delaunoy et al. (2020) make inference on gravitational waves; and Cole et al. (2022) infer cosmological parameters based on measurements of the cosmic microwave background.

S3 ADDITIONAL TABLES AND FIGURES

Table S1: Outline of software and architectures used in the illustration of Section 6.2

Acronym	Method	Implementation
MCMC	Metropolis–Hastings.	Base R; R Core Team (2023).
NBE	Neural Bayes estimation.	NeuralEstimators ; Sainsbury-Dale et al. (2024). Summary network: two-layer CNN followed by two fully-connected layers. Inference network: two fully-connected layers to output posterior mean and 0.05/0.95 posterior quantiles.
fKL	Forward KL minimization with normalizing flow approximate posterior.	BayesFlow ; Radev et al. (2022). Summary network: two-layer CNN followed by two fully-connected layers. Inference network: Invertible neural network with four coupling layers to sample from the posterior distribution.
rKL1	Reverse KL minimization with logit-Gaussian approximate posterior.	TensorFlow ; Abadi et al. (2016). Inference network: two-layer CNN followed by two fully-connected layers to output variational mean and variance.
rKL2	Reverse KL minimization with logit-Gaussian approximate posterior and with Gaussian synthetic likelihood constructed using one “expert” summary statistic.	TensorFlow ; Abadi et al. (2016). Binding function network: two fully-connected layers that output mean and variance of summary statistic. Inference network: two-layer CNN followed by two fully-connected layers to output variational mean and variance.
rKL3	Reverse KL minimization with logit-Gaussian approximate posterior and with Gaussian synthetic likelihood constructed using one summary statistic found by maximizing mutual information.	TensorFlow ; Abadi et al. (2016). Summary network: two-layer CNN followed by two fully-connected layers to output summary statistic from input data. Binding function network: two fully-connected layers that output mean and variance of summary statistic. Inference network: two-layer CNN followed by two fully-connected layers to output variational mean and variance.
NRE	Neural ratio estimation with sampling from the posterior distribution on a fine gridding of the parameter space.	sbi ; Tejero-Cantero et al. (2020) Summary network: two-layer CNN followed by two fully-connected layers. Inference network: three fully-connected layers to output likelihood-to-evidence ratio.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals,

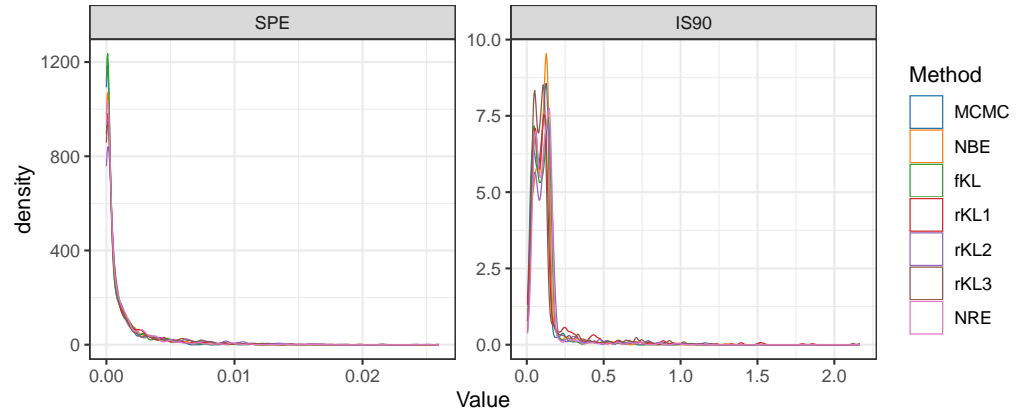


Figure S3: Empirical distribution of the squared prediction errors (left) and the 90% interval scores (right) for the different methods considered in the illustration of Section 6.2.

- O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2016). TensorFlow: Large-scale machine learning on heterogeneous systems. *arXiv:1603.04467* [cs.DC].
- Ahmed, M., Maume-Deschamps, V., and Ribereau, P. (2022). Recognizing a spatial extreme dependence structure: A deep learning approach. *Environmetrics*, 33:e2714.
- Baldi, P., Cranmer, K., Faucett, T., Sadowski, P., and Whiteson, D. (2016). Parameterized neural networks for high-energy physics. *The European Physical Journal C*, 76:235.
- Bishop, C. (1995). *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, UK.
- Bissiri, P. G., Holmes, C. C., and Walker, S. G. (2016). A general framework for updating belief distributions. *Journal of the Royal Statistical Society B*, 78:1103–1130.
- Boelts, J., Lueckmann, J.-M., Gao, R., and Macke, J. H. (2022). Flexible and efficient simulation-based inference for models of decision-making. *eLife*, 11:e77220.
- Bon, J. J., Bretherton, A., Buchhorn, K., Cramb, S., Drovandi, C., Hassan, C., Jenner, A. L., Mayfield, H. J., McGree, J. M., Mengersen, K., et al. (2023). Being Bayesian in the 2020s: opportunities and challenges in the practice of modern applied Bayesian statistics. *Philosophical Transactions of the Royal Society A*, 381:20220156.
- Brehmer, J., Louppe, G., Pavez, J., and Cranmer, K. (2020). Mining gold from implicit models to improve likelihood-free inference. *Proceedings of the National Academy of Sciences*, 117:5242–5249.
- Cartwright, L., Zammit-Mangion, A., and Deutscher, N. M. (2023). Emulation of greenhouse-gas sensitivities using variational autoencoders. *Environmetrics*, 34:e2754.
- Castruccio, S., Huser, R., and Genton, M. G. (2016). High-order composite likelihood inference for max-stable distributions and processes. *Journal of Computational and Graphical Statistics*, 25:1212–1229.
- Chan, J., Perrone, V., Spence, J., Jenkins, P., Mathieson, S., and Song, Y. (2018). A likelihood-free inference framework for population genetic data using exchangeable neural networks. In *Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS 2018)*, pages 8594–8605, Red Hook, NY. Curran.

- Cole, A., Miller, B. K., Witte, S. J., Cai, M. X., Grootes, M. W., Nattino, F., and Weniger, C. (2022). Fast and credible likelihood-free cosmology with truncated marginal neural ratio estimation. *Journal of Cosmology and Astroparticle Physics*, 2022:004.
- Dalmaso, N., Masserano, L., Zhao, D., Izbicki, R., and Lee, A. B. (2021). Likelihood-free frequentist inference: Bridging classical statistics and machine learning for reliable simulator-based inference. arXiv:2107.03920 [stat.ML].
- David, L., Bréon, F.-M., and Chevallier, F. (2021). XCO₂ estimates from the OCO-2 measurements using a neural network approach. *Atmospheric Measurement Techniques*, 14:117–132.
- Davison, A. C. (2003). *Statistical Models*. Cambridge University Press, Cambridge, UK.
- Davison, A. C. and Huser, R. (2015). Statistics of extremes. *Annual Review of Statistics and its Application*, 2:203–235.
- Davison, A. C., Huser, R., and Thibaud, E. (2019). Spatial extremes. In Gelfand, A. E., Fuentes, M., Hoeting, J. A., and Smith, R. L., editors, *Handbook of Environmental and Ecological Statistics*, pages 711–744. Chapman & Hall/CRC Press, Boca Raton, FL.
- Davison, A. C., Padoan, S. A., and Ribatet, M. (2012). Statistical modeling of spatial extremes (with discussion). *Statistical Science*, 27:161–201.
- de Bézenac, E., Pajot, A., and Gallinari, P. (2018). Deep learning for physical processes: Incorporating prior scientific knowledge. Vancouver, Canada: OpenReview. <https://openreview.net/forum?id=By4HsfWAZ>.
- Delaunoy, A., Wehenkel, A., Hinderer, T., Nissanke, S., Weniger, C., Williamson, A. R., and Louppe, G. (2020). Lightning-fast gravitational wave parameter inference through neural amortization. arXiv:2010.12931 [astro-ph.IM].
- Dinev, T. and Gutmann, M. U. (2018). Dynamic likelihood-free inference via ratio estimation (DIRE). arXiv:1810.09899 [stat.ML].
- Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2016). Density estimation using Real NVP. In *Proceedings of the 4th International Conference on Learning Representations (ICLR 2016)*. San Juan, Puerto Rico: OpenReview. <https://openreview.net/forum?id=HkpbnH91x>.
- Doersch, C. (2016). Tutorial on variational autoencoders. arXiv:1606.05908 [stat.ML].
- Fengler, A., Govindarajan, L., Chen, T., and Frank, M. J. (2021). Likelihood approximation networks (LANs) for fast inference of simulation models in cognitive neuroscience. *eLife*, 10:e65074.
- Gerber, F. and Nychka, D. (2021). Fast covariance parameter estimation of spatial Gaussian process models using neural networks. *Stat*, 10:e382.
- Glöckler, M., Deistler, M., and Macke, J. H. (2022). Variational methods for simulation-based inference. In *Proceedings of the 10th International Conference on Learning Representations (ICLR 2022)*. Virtual: OpenReview. <https://openreview.net/forum?id=kZOUYdhqkNY>.
- Gneiting, T., Balabdaoui, F., and Raftery, A. E. (2007). Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society B*, 69:243–268.
- Goh, H., Sherifdeen, S., Wittmer, J., and Bui-Thanh, T. (2019). Solving Bayesian inverse problems via variational autoencoders. In *Proceedings of the 2nd Mathematical and Scientific Machine Learning Conference (MSML 2022)*, volume 145, pages 386–425. PMLR.

- Goncalves, P. J., Lueckmann, J.-M., Deistler, M., Nonnenmacher, M., Ocal, K., Bassetto, G., Chintaluri, C., Podlaski, W. F., Haddad, S. A., Vogels, T. P., et al. (2020). Training deep neural density estimators to identify mechanistic models of neural dynamics. *eLife*, 9:e56261.
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Proceedings of the 28th Conference on Neural Information Processing Systems (NeurIPS 2014)*, pages 2672–2680, Red Hook, NY. Curran.
- Greenberg, D., Nonnenmacher, M., and Macke, J. (2019). Automatic posterior transformation for likelihood-free inference. In *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)*, volume 97, pages 2404–2414. PMLR.
- Hermans, J., Delaunoy, A., Rozet, F., Wehenkel, A., Begy, V., and Louppe, G. (2022). A crisis in simulation-based inference? Beware, your posterior approximations can be unfaithful. *Transactions on Machine Learning Research*. OpenReview. <https://openreview.net/pdf?id=LHAbHkt6Aq>.
- Hjelm, D., Salakhutdinov, R. R., Cho, K., Jovic, N., Calhoun, V., and Chung, J. (2016). Iterative refinement of the approximate posterior for directed belief networks. In *Proceedings of the 30th Conference on Neural Information Processing Systems (NeurIPS 2016)*, pages 4698–4706, Red Hook, NY. Curran.
- Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9:1735–1780.
- Huber, P. J. (1964). Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 53:73–101.
- Huser, R., Opitz, T., and Wadsworth, J. L. (2024). Modeling of spatial extremes in environmental data science: Time to move away from max-stable processes. [arXiv:2401.17430 \[stat.ME\]](https://arxiv.org/abs/2401.17430).
- Huser, R. and Wadsworth, J. L. (2022). Advances in statistical modeling of spatial extremes. *Wiley Interdisciplinary Reviews: Computational Statistics*, 14:e1537.
- Katzfuss, M. and Guinness, J. (2021). A general framework for Vecchia approximations of Gaussian processes. *Statistical Science*, 36:124–141.
- Kelly, R. P., Nott, D. J., Frazier, D. T., Warne, D. J., and Drovandi, C. (2023). Misspecification-robust sequential neural likelihood. [arXiv:2301.13368 \[stat.ME\]](https://arxiv.org/abs/2301.13368).
- Kim, Y., Wiseman, S., Miller, A., Sontag, D., and Rush, A. (2018). Semi-amortized variational autoencoders. In *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, volume 80, pages 2678–2687. PMLR.
- Kingma, D. P. and Welling, M. (2013). Auto-encoding variational Bayes. [arXiv:1312.6114 \[stat.ML\]](https://arxiv.org/abs/1312.6114).
- Kingma, D. P., Welling, M., et al. (2019). An introduction to variational autoencoders. *Foundations and Trends in Machine Learning*, 12:307–392.
- Kobyzev, I., Prince, S. J., and Brubaker, M. A. (2020). Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43:3964–3979.
- Kruse, J., Detommaso, G., Köthe, U., and Scheichl, R. (2021). HINT: Hierarchical invertible neural transport for density estimation and Bayesian inference. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI-21)*, volume 35, pages 8191–8199. AAAI.

- LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278–2324.
- Lenzi, A., Bessac, J., Rudi, J., and Stein, M. L. (2023). Neural networks for parameter estimation in intractable models. *Computational Statistics & Data Analysis*, 185:107762.
- Lueckmann, J.-M., Bassetto, G., Karaletsos, T., and Macke, J. H. (2019). Likelihood-free inference with emulator networks. In *Proceedings of The 1st Symposium on Advances in Approximate Bayesian Inference*, volume 96, pages 32–53. PMLR.
- Lueckmann, J.-M., Goncalves, P. J., Bassetto, G., Öcal, K., Nonnenmacher, M., and Macke, J. H. (2017). Flexible statistical inference for mechanistic models of neural dynamics. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS 2017)*, pages 1290–1300, Red Hook, NY. Curran.
- Majumder, R. and Reich, B. (2023). A deep learning synthetic likelihood approximation of a non-stationary spatial model for extreme streamflow forecasting. *Spatial Statistics*, 55:100755.
- Majumder, R., Reich, B., and Shaby, B. (2023). Modeling extremal streamflow using deep learning approximations and a flexible spatial process. *Annals of Applied Statistics*, in press.
- Masserano, L., Dorigo, T., Izbicki, R., Kuusela, M., and Lee, A. B. (2023). Simulator-based inference with WALDO: Confidence regions by leveraging prediction algorithms and posterior estimators for inverse problems. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS 2023)*, volume 206, pages 2960–2974. PMLR.
- Pacchiardi, L. and Dutta, R. (2022). Likelihood-free inference with generative neural networks via scoring rule minimization. arXiv:2205.15784 [stat.CO].
- Padoan, S. A., Ribatet, M., and Sisson, S. A. (2010). Likelihood-based inference for max-stable processes. *Journal of the American Statistical Association*, 105:263–277.
- Papamakarios, G. and Murray, I. (2016). Fast ε -free inference of simulation models with Bayesian conditional density estimation. In *Proceedings of the 30th Conference on Neural Information Processing Systems (NeurIPS 2016)*, pages 1036–1044, Red Hook, NY. Curran.
- Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2021). Normalizing flows for probabilistic modeling and inference. *The Journal of Machine Learning Research*, 22:2617–2680.
- Papamakarios, G., Pavlakou, T., and Murray, I. (2017). Masked autoregressive flow for density estimation. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS 2017)*, pages 2339–2348, Red Hook, NY. Curran.
- Papamakarios, G., Sterratt, D., and Murray, I. (2019). Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS 2019)*, volume 89, pages 837–848. PMLR.
- Radev, S. T., Mertens, U. K., Voss, A., Ardizzone, L., and Köthe, U. (2022). BayesFlow: Learning complex stochastic models with invertible neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 33:1452–1466.

- Radev, S. T., Schmitt, M., Pratz, V., Picchini, U., Köethe, U., and Buerkner, P.-C. (2023). JANA: Jointly amortized neural approximation of complex Bayesian models. In *Proceedings of the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023)*, volume 216, pages 1695–1706. PMLR.
- Ramesh, P., Lueckmann, J.-M., Boelts, J., Tejero-Cantero, Á., Greenberg, D. S., Goncalves, P. J., and Macke, J. H. (2022). GATSBI: Generative adversarial training for simulation-based inference. In *Proceedings of the 10th International Conference on Learning Representations (ICLR 2022)*. Virtual: OpenReview. <https://openreview.net/forum?id=kR1hC6j48Tp>.
- R Core Team (2023). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.
- Richards, J., Sainsbury-Dale, M., Zammit-Mangion, A., and Huser, R. (2023). Likelihood-free neural Bayes estimators for censored peaks-over-threshold models. arXiv:2306.15642 [stat.ME].
- Rudi, J., Bessac, J., and Lenzi, A. (2022). Parameter estimation with dense and convolutional neural networks applied to the FitzHugh–Nagumo ODE. In *Proceedings of the 2nd Mathematical and Scientific Machine Learning Conference (MSML 2022)*, volume 145, pages 781–808. PMLR.
- Sainsbury-Dale, M., Richards, J., Zammit-Mangion, A., and Huser, R. (2023). Neural Bayes estimators for irregular spatial data using graph neural networks. arXiv:2310.02600 [stat.ME].
- Sainsbury-Dale, M., Zammit-Mangion, A., and Huser, R. (2024). Likelihood-free parameter estimation with neural Bayes estimators. *The American Statistician*, 78:1–14.
- Schlather, M. (2002). Models for stationary max-stable random fields. *Extremes*, 5:33–44.
- Schmitt, M., Bürkner, P.-C., Köthe, U., and Radev, S. T. (2024). Detecting model misspecification in amortized Bayesian inference with neural networks. In Köthe, U. and Rother, C., editors, *Pattern Recognition. DAGM GCPR 2023*, pages 541–557, Cham, Switzerland. Springer.
- Siakhoochi, A., Rizzuti, G., Orozco, R., and Herrmann, F. J. (2023). Reliable amortized variational inference with physics-based latent distribution correction. *Geophysics*, 88:297–322.
- Speiser, A., Yan, J., Archer, E. W., Buesing, L., Turaga, S. C., and Macke, J. H. (2017). Fast amortized inference of neural activity from calcium imaging data with variational autoencoders. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS 2017)*, pages 4025–4035, Red Hook, NY. Curran.
- Stein, M. L., Chi, Z., and Welty, L. J. (2004). Approximating likelihoods for large spatial data sets. *Journal of the Royal Statistical Society B*, 66:275–296.
- Talts, S., Betancourt, M., Simpson, D., Vehtari, A., and Gelman, A. (2018). Validating Bayesian inference algorithms with simulation-based calibration. arXiv:1804.06788 [stat.ME].
- Tejero-Cantero, A., Boelts, J., Deistler, M., Lueckmann, J.-M., Durkan, C., Goncalves, P. J., Greenberg, D. S., and Macke, J. H. (2020). sbi: A toolkit for simulation-based inference. *Journal of Open Source Software*, 5:2505.

- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS 2017)*, pages 5999–6009, Red Hook, NY. Curran.
- Vecchia, A. V. (1988). Estimation and model identification for continuous spatial processes. *Journal of the Royal Statistical Society B*, 50:297–312.
- von Krause, M., Radev, S. T., and Voss, A. (2022). Mental speed is high until age 60 as revealed by analysis of over a million participants. *Nature Human Behaviour*, 6:700–708.
- Wadsworth, J. L. and Tawn, J. A. (2012). Dependence modelling for spatial extremes. *Biometrika*, 99:253–272.
- Wang, Z., Hasenauer, J., and Schälte, Y. (2024). Missing data in amortized simulation-based neural posterior estimation. *PLOS Computational Biology*, 20:e1012184.
- Williams, R. J. and Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1:270–280.
- Wiqvist, S., Frellsen, J., and Picchini, U. (2021). Sequential neural posterior and likelihood approximation. arXiv:2102.06522 [stat.ML].
- Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. (2021). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32:4–24.
- Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. (2017). Deep sets. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS 2017)*, pages 3392–3402, Red Hook, NY. Curran.
- Zammit-Mangion, A. and Wikle, C. K. (2020). Deep integro-difference equation models for spatio-temporal forecasting. *Spatial Statistics*, 37:100408.
- Zhang, S., Tong, H., Xu, J., and Maciejewski, R. (2019). Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6:11.
- Zhao, D., Dalmaso, N., Izbicki, R., and Lee, A. B. (2021). Diagnostics for conditional density models and Bayesian inference algorithms. In *Proceedings of the 37th Conference on Uncertainty in Artificial Intelligence (UAI 2021)*, volume 161, pages 1830–1840. PMLR.
- Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. (2020). Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81.
- Åkesson, M., Singh, P., Wrede, F., and Hellander, A. (2022). Convolutional neural networks as summary statistics for approximate Bayesian computation. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 19:3353–3365.