Leveraging Self-Consistency for Data-Efficient Amortized Bayesian Inference

Marvin Schmitt [1] **Desi R. Ivanova** [2] **Daniel Habermann** [3] **Ullrich Köthe** [4] **Paul-Christian Bürkner** [3]
Stefan T. Radev [5]

Abstract

We propose a method to improve the efficiency and accuracy of amortized Bayesian inference by leveraging universal symmetries in the joint probabilistic model $p(\boldsymbol{\theta}, \mathbf{Y})$ of parameters $\boldsymbol{\theta}$ and data $\mathbf{Y}$. In a nutshell, we invert Bayes' theorem and estimate the marginal likelihood based on approximate representations of the joint model. Upon perfect approximation, the marginal likelihood is constant across all parameter values by definition. However, errors in approximate inference lead to undesirable variance in the marginal likelihood estimates across different parameter values. We penalize violations of this symmetry with a *self-consistency loss* which significantly improves the quality of approximate inference in low data regimes and can be used to augment the training of popular neural density estimators. We apply our method to a number of synthetic problems and realistic scientific models, discovering notable advantages in the context of both neural posterior and likelihood approximation.

1. Introduction

Computer simulations are ubiquitous in today's world, and their widespread application in the sciences has heralded a new era of *simulation intelligence* (Lavin et al., 2021). Typically, scientific simulators define a mapping from latent parameters $\boldsymbol{\theta}$ to observable data $\mathbf{Y}$. This *forward problem* is probabilistically described by the likelihood $p(\mathbf{Y} \,|\, \boldsymbol{\theta})$. The *inverse problem* of reasoning about the unknown parameters $\boldsymbol{\theta}$ given observed data $\mathbf{Y}$ and a prior $p(\boldsymbol{\theta})$ is captured by the posterior $p(\boldsymbol{\theta} \,|\, \mathbf{Y}) = p(\boldsymbol{\theta})\, p(\mathbf{Y} \,|\, \boldsymbol{\theta}) / p(\mathbf{Y})$, which represents a coherent way to combine all available information in a probabilistic system (Gelman et al., 2013) and quantify epistemic uncertainty (Hüllermeier & Waegeman, 2021). For complex models, the marginal likelihood $p(\mathbf{Y})$ is a high-dimensional integral, $p(\mathbf{Y}) = \int p(\boldsymbol{\theta})\, p(\mathbf{Y} \,|\, \boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta}$, rendering the posterior analytically intractable in general.

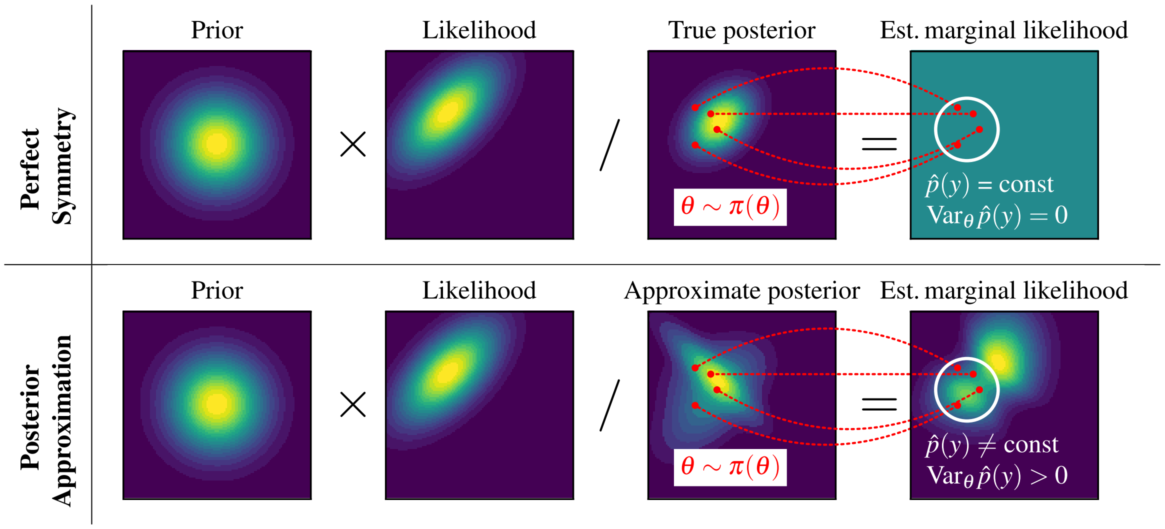
In *likelihood-based inference*, the likelihood is explicitly available as a probability density function $p(\mathbf{Y} \,|\, \boldsymbol{\theta})$, giving rise to a family of likelihood-based algorithms to approximate the posterior distribution. Markov chain Monte Carlo (MCMC) methods sample from the unnormalized posterior by exploring the parameter space through a Markov chain, with state-of-the-art samplers such as Hamiltonian Monte Carlo (Neal, 2011), as implemented in the probabilistic programming language Stan (Carpenter et al., 2017). Variational inference approximates the posterior via tractable analytic distributions, with consistent progress towards more trustworthy variational methods (Blei et al., 2017).

Different from likelihood-based inference, *simulation-based inference* (SBI) circumvents explicit likelihood evaluation and relies only on random samples from a simulation program $\mathbf{Y} \sim p(\mathbf{Y}, \mathbf{Z} \,|\, \boldsymbol{\theta})$ with latent program states or "outsourced" noise $\mathbf{Z}$ (Cranmer et al., 2020). The execution paths of the simulation program define an implicit likelihood $p(\mathbf{Y} \,|\, \boldsymbol{\theta}) = \int p(\mathbf{Y}, \mathbf{Z} \,|\, \boldsymbol{\theta}) \mathrm{d}\mathbf{Z}$, which is computationally intractable for any simulation program of practical interest. However, we have access to samples $(\boldsymbol{\theta}, \mathbf{Y})$ of parameter-data tuples by executing the simulation program repeatedly. In the face of analytically intractable simulators, previous research has explored other properties of such programs for learning surrogate likelihood functions or the likelihood ratio (Brehmer et al., 2018; 2020b;a).

As a general perspective on simulation intelligence, *amortized Bayesian inference* (ABI) is concerned with enabling fully probabilistic SBI in real-time (Radev et al., 2020; Gonçalves et al., 2020; Avecilla et al., 2022). A core principle of ABI lies in tackling probabilistic problems (forward, inverse, or both) with neural networks. By re-casting an intractable probabilistic problem as forward passes through a trained generative neural network, the required computational time reduces from hours (MCMC) to just a few seconds (ABI). Yet, there is rarely a free lunch, and neural ABI algorithms require an upfront training phase. The associated effort is subsequently repaid with real-time inference on new data sets, thereby *amortizing* the initial training time.

[1]University of Stuttgart, Germany [2]University of Oxford, UK [3]TU Dortmund University, Germany [4]Heidelberg University, Germany [5]Rensselaer Polytechnic Institute, USA. Correspondence to: Marvin Schmitt <mail.marvinschmitt@gmail.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).



Perfect Symmetry
Prior
Likelihood
True posterior
Est. marginal likelihood
$\times$
$/$
$=$
$\theta \sim \pi(\theta)$
$\hat{p}(y) = \text{const}$
$\text{Var}_\theta \hat{p}(y) = 0$
Posterior Approximation
Prior
Likelihood
Approximate posterior
Est. marginal likelihood
$\theta \sim \pi(\theta)$
$\hat{p}(y) \neq \text{const}$
$\text{Var}_\theta \hat{p}(y) > 0$

Figure 1: The performance of the posterior approximator is evaluated via the variance of the corresponding marginal likelihood estimates. **Top row:** For the true posterior (or a perfect approximation thereof), the estimated marginal likelihood is constant for any parameter value $\boldsymbol{\theta} \sim \pi(\boldsymbol{\theta})$. **Bottom row:** For an imperfect approximate posterior, the estimated marginal likelihood varies across different parameter values. Hence, the inherent symmetry of the joint probabilistic model $p(\boldsymbol{\theta}, \mathbf{Y})$ is violated by its approximate representation. Minimizing the variance of the marginal likelihood estimates pushes the estimated marginal likelihood towards uniformity. This restores the symmetry of the unified representation, which is equivalent to improving the approximate posterior.

In this paper, we propose a new method to improve the accuracy of ABI, particularly in low data regimes. We achieve this by leveraging universal symmetries in the joint probabilistic model $p(\boldsymbol{\theta}, \mathbf{Y})$. In a nutshell, we invert Bayes' theorem and estimate the marginal likelihood based on an approximate likelihood and posterior. Upon perfect approximation, the estimated marginal likelihood is constant across all parameters. An imperfect approximation, however, leads to variance in the marginal likelihood estimates across different underlying parameter values (see Figure 1). We penalize violations of this symmetry in the loss function to accelerate the training of neural density estimators that approximate components of the probabilistic model. We apply our novel method to a range of synthetic and real problems with both an explicit likelihood (likelihood-based) and an implicit likelihood (simulation-based). Our contributions are:

(i) We propose a novel *self-consistency loss* which exploits symmetries of the joint probabilistic model $p(\boldsymbol{\theta}, \mathbf{Y})$ in neural network representations of its components;

(ii) We increase the training efficiency of amortized neural posterior estimation by leveraging information from an explicit likelihood for the self-consistency loss;

(iii) We demonstrate how simultaneous learning of an approximate posterior and a surrogate likelihood benefits from the self-consistency loss without requiring an explicit likelihood.

2. Background

2.1. Notation

Simulation-based training uses a *training set* $\{(\boldsymbol{\theta}^{(i)}, \mathbf{Y}^{(i)})\}_{i=1}^{N}$ of tuples of simulated parameters and data. Here, N is the total *simulation budget* for neural network training, and each superscript i marks one training example (i.e., tuple of latent parameter vector and observable data set). In accordance with the (Bayesian) forward model, we summarize the D-dimensional latent parameter vector of the simulation program as $\boldsymbol{\theta} \equiv (\theta_1, \dots, \theta_D)$. Further, $\mathbf{Y} \equiv \{\mathbf{y}_j\}_{j=1}^{J}$ is a data set, that is, a matrix whose rows consist of multi-dimensional (vector-valued) observations $\{\mathbf{y}_j\}_{j=1}^{J} \equiv \{\mathbf{y}_1, \dots, \mathbf{y}_J\}$. Accordingly, one parameter vector $\boldsymbol{\theta}^{(i)}$ yields one data set $\mathbf{Y}^{(i)}$.

2.2. Neural Posterior Estimation

Past work on neural SBI has primarily focused on neural posterior estimation (NPE) for cases where no explicit likelihood is available (Radev et al., 2020; Gonçalves et al., 2020; Avecilla et al., 2022; Geffner et al., 2023; Sharrock et al., 2022; Schmitt et al., 2023a). NPE typically builds on a conditional normalizing flow (Rezende & Mohamed, 2015) $f_{\boldsymbol{\phi}}(\boldsymbol{\theta}; \mathbf{Y})$ with trainable neural network weights $\boldsymbol{\phi}$. It implements a bijective map between the inference targets $\boldsymbol{\theta} \in \mathbb{R}^D$ and a latent variable $\mathbf{z} \in \mathbb{R}^D$ with a simple base distribution $p(\mathbf{z})$, e.g., Gaussian or Student-t. The normalizing flow defines a probability distribution $q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \mathbf{Y})$ through change-of-variables,

$$q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \mathbf{Y}) = p(\mathbf{z} = f_{\boldsymbol{\phi}}(\boldsymbol{\theta}; \mathbf{Y})) \left| \det \frac{\partial f_{\boldsymbol{\phi}}(\boldsymbol{\theta}; \mathbf{Y})}{\partial \boldsymbol{\theta}} \right|, \quad (1)$$

resulting in a direct surrogate for the target posterior distribution $p(\boldsymbol{\theta} \mid \mathbf{Y})$. The neural network weights $\boldsymbol{\phi}$ are trained by minimizing the forward Kullback-Leibler (KL) divergence between the target posterior and approximate posterior:

$$\begin{aligned} \mathcal{L}_{\text{NPE}}(\boldsymbol{\phi}) &= \mathbb{E}_{p(\mathbf{Y})}\left[\mathrm{KL}(p(\boldsymbol{\theta} \mid \mathbf{Y}) \,\|\, q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \mathbf{Y}))\right] \\ &= \mathbb{E}_{p(\boldsymbol{\theta}, \mathbf{Y})}[-\log q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \mathbf{Y})] + \text{const} \end{aligned} \quad (2)$$

Since NPE algorithms are trained across the entire prior predictive distribution $p(\mathbf{Y})$, they naturally amortize over multiple new data sets during inference. In **Experiments 1** and **4**, we apply NPE to cases where we *have* access to an explicit likelihood but still want to achieve amortized inference, which is infeasible with MCMC-based algorithms.

2.3. Neural Posterior and Likelihood Estimation

Recently, simulation-based inference and surrogate modeling have been tackled *jointly* by learning an approximate posterior $q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \mathbf{Y})$ and a surrogate likelihood $q_{\boldsymbol{\eta}}(\mathbf{Y} \mid \boldsymbol{\theta})$ in tandem. This approach is called neural posterior and likelihood estimation (NPLE; Wiqvist et al., 2021; Glöckler et al., 2022; Radev et al., 2023). Amortized NPLE (e.g., Radev et al., 2023) combines the maximum likelihood objectives for posterior and likelihood into a joint loss:

$$\mathcal{L}_{\text{NPLE}}(\boldsymbol{\phi}, \boldsymbol{\eta}) = \mathbb{E}_{p(\boldsymbol{\theta}, \mathbf{Y})}[-\log q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \mathbf{Y}) - \log q_{\boldsymbol{\eta}}(\mathbf{Y} \mid \boldsymbol{\theta})] \quad (3)$$

NPLE enables the joint estimation of both amortized posterior predictive and marginal likelihood. This expands the utility of Bayesian workflows to downstream tasks that have historically been deemed computationally impractical, such as cross-validation with likelihood-based predictive metrics (Radev et al., 2023; Vehtari et al., 2022).

2.4. Limitations of NPE and NPLE

Whilst NPE and NPLE have shown promise in complex applications, they often require large simulation budgets and do not explicitly encourage accurate marginal likelihood estimation. In this paper, we propose a straightforward yet powerful approach to improve amortized Bayesian inference, integrating a self-consistency mechanism that alleviates these issues and enhances the accuracy of posterior and likelihood estimation with small simulation budgets.

3. Leveraging Self-Consistency for ABI

The joint model $p(\boldsymbol{\theta}, \mathbf{Y})$ implies a symmetry between marginal likelihood $p(\mathbf{Y})$, prior $p(\boldsymbol{\theta})$, likelihood $p(\mathbf{Y} \mid \boldsymbol{\theta})$, and posterior $p(\boldsymbol{\theta} \mid \mathbf{Y})$. Inverting Bayes' theorem yields

$$p(\mathbf{Y}) = \frac{p(\boldsymbol{\theta})\, p(\mathbf{Y} \mid \boldsymbol{\theta})}{p(\boldsymbol{\theta} \mid \mathbf{Y})}, \quad (4)$$

which must still hold if any component of the joint model is represented through a *perfect* approximator $q(\cdot)$. However, we cannot directly use Eq. 4 as a loss function for learning the posterior because the marginal likelihood on the LHS is notoriously difficult to approximate with high precision (Meng & Wong, 1996). Instead, we exploit the fact that $p(\mathbf{Y})$ is constant across all parameters $\boldsymbol{\theta}$ (LHS), even though its computation (RHS) hinges on an arbitrary but fixed parameter value $\boldsymbol{\theta}$. In other words, if we choose K parameter values $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K$, all computed marginal likelihood values must be *equal*, regardless of the individual parameter $\boldsymbol{\theta}_k$. We call this the *self-consistency criterion*,

$$\begin{aligned} \frac{p(\boldsymbol{\theta})\, p(\mathbf{Y} \mid \boldsymbol{\theta})}{p(\boldsymbol{\theta} \mid \mathbf{Y})} &= \text{const } \forall \boldsymbol{\theta} \in \Theta \implies \\ \frac{p(\boldsymbol{\theta}_1)\, p(\mathbf{Y} \mid \boldsymbol{\theta}_1)}{p(\boldsymbol{\theta}_1 \mid \mathbf{Y})} &= \ldots = \frac{p(\boldsymbol{\theta}_K)\, p(\mathbf{Y} \mid \boldsymbol{\theta}_K)}{p(\boldsymbol{\theta}_K \mid \mathbf{Y})}, \end{aligned} \quad (5)$$

where Θ denotes the admissable parameter space and $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K \in \Theta$ are arbitrary but fixed parameter values. In this paper, the self-consistency criterion shall be applied to neural posterior and likelihood estimation. We will first demonstrate that direct constrained optimization is computationally infeasible. Then, we will propose a straightforward and robust way to integrate the self-consistency criterion into existing neural density estimators.

3.1. Naïve Approach: Direct Constrained Optimization

We could optimize the NPE (2) or NPLE (3) objective, subject to the self-consistency constraint (5), through Lagrange multipliers. For (2), this would involve optimizing

$$\begin{aligned} L(\boldsymbol{\phi}, \lambda_{1:K}) &= \mathcal{L}_{\text{NPE}}(\boldsymbol{\phi}) + \mathcal{L}(\boldsymbol{\phi}, \lambda_{1:K}), \quad \text{where} \\ \mathcal{L}(\boldsymbol{\phi}, \lambda_{1:K}) &:= \sum_{k=1}^{K} \lambda_k \left(\frac{p(\boldsymbol{\theta}_k) p(\mathbf{Y} \mid \boldsymbol{\theta}_k)}{q_\phi(\boldsymbol{\theta}_k \mid \mathbf{Y})} - c \right) \end{aligned} \quad (6)$$

with respect to $\boldsymbol{\phi}$ and $\lambda_{1:K}$. The inconvenience of the unknown constant c could be resolved by formulating a relative constraint, that is, choosing

$$\mathcal{L}(\boldsymbol{\phi}, \lambda_{1:K}) := \sum_{k=1}^{K-1} \lambda_k \left(\frac{p(\boldsymbol{\theta}_k) p(\mathbf{Y} \mid \boldsymbol{\theta}_k)}{q_\phi(\boldsymbol{\theta}_k \mid \mathbf{Y})} - \frac{p(\boldsymbol{\theta}_{k+1}) p(\mathbf{Y} \mid \boldsymbol{\theta}_{k+1})}{q_\phi(\boldsymbol{\theta}_{k+1} \mid \mathbf{Y})} \right). \quad (7)$$

While this approach is conceptually appealing, such relative constraints are notoriously hard to optimize in practice, which calls for an alternative solution that is more scalable.

Algorithm 1 Self-consistency loss for finite training.
{I}: likelihood-based with analytic likelihood
{II}: simulation-based with approximate likelihood

```
Input: $N$ training data tuples $\{(\boldsymbol{\theta}^{(i)}, \mathbf{Y}^{(i)})\}_{i=1}^N$
Input: Number of self-consistency samples $K$
1: for $i = 1, \ldots, N$ do
2:     < compute other losses such as NPE/NPLE loss >
3:     for $k = 1, \ldots, K$ do
4:         $\boldsymbol{\theta}_k \sim \pi(\boldsymbol{\theta})$    {sample from proposal $\pi(\boldsymbol{\theta})$}
5:         $\log \hat{p}_k(\mathbf{Y}^{(i)}) = \begin{cases} \log \dfrac{p(\boldsymbol{\theta}_k)\, p(\mathbf{Y}^{(i)} \mid \boldsymbol{\theta}_k)}{q_{\boldsymbol{\phi}}(\boldsymbol{\theta}_k \mid \mathbf{Y}^{(i)})} & \text{\{I\}} \\ \log \dfrac{p(\boldsymbol{\theta}_k)\, q_{\boldsymbol{\eta}}(\mathbf{Y}^{(i)} \mid \boldsymbol{\theta}_k)}{q_{\boldsymbol{\phi}}(\boldsymbol{\theta}_k \mid \mathbf{Y}^{(i)})} & \text{\{II\}} \end{cases}$
6:     end for
7:     $\mathcal{L}_{SC}^{(i)} = \mathrm{Var}(\{\log \hat{p}_k(\mathbf{Y}^{(i)})\}_{k=1}^K)$
8: end for
```

3.2. Variance Penalty and Self-Consistency Loss

To overcome the problems of the naïve approach, we reframe the constraint as a variance penalty, which simplifies the optimization process and enhances both computational feasibility and robustness compared to direct second order optimization. First, Köthe (2023) observes that a second-order Taylor expansion of Eq. 2 yields

$$\mathrm{KL}\big(p(\boldsymbol{\theta} \mid \mathbf{Y}) \,\|\, q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \mathbf{Y})\big) \approx \frac{1}{2p(\mathbf{Y})^2} \mathrm{Var}_{p(\boldsymbol{\theta} \mid \mathbf{Y})} \left(\frac{p(\boldsymbol{\theta})\, p(\mathbf{Y} \mid \boldsymbol{\theta})}{q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \mathbf{Y})} \right). \tag{8}$$

This approximation implies that the KL divergence between the true and the approximate posterior becomes negligible as the variance of Eq. 4 with respect to the true posterior $p(\boldsymbol{\theta} \mid \mathbf{Y})$ shrinks to zero. Moreover, it is clear that minimizing the variance of the marginal likelihood estimator indirectly achieves the effect of the constraint implied by Eq. 7. However, directly targeting the variance introduces two new challenges: (i) The true posterior $p(\boldsymbol{\theta} \mid \mathbf{Y})$ is unknown; and (ii) the argument of $\mathrm{Var}_{p(\boldsymbol{\theta} \mid \mathbf{Y})}[\cdot]$ may cause numerical instabilities due to the danger of vanishingly small values in the denominator.

As a remedy, we propose a two-step solution during the simulation-based training: (i) Sample parameters $\boldsymbol{\theta}$ from a proposal distribution $\pi(\boldsymbol{\theta})$; and (ii) quantify the expected violation of the self-consistency criterion via the variance of the estimated *log* marginal likelihood (LML) across these parameter samples. This results in the following *self-consistency loss function*:

$$\mathcal{L}_{\mathrm{SC}}(\mathbf{Y}, \boldsymbol{\phi}) = \mathrm{Var}_{\pi(\boldsymbol{\theta})} \left(\log \frac{p(\boldsymbol{\theta})\, p(\mathbf{Y} \mid \boldsymbol{\theta})}{q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \mathbf{Y})} \right) \tag{9}$$

The analytic likelihood $p(\mathbf{Y} \mid \boldsymbol{\theta})$ may be replaced with an approximate likelihood $q_{\boldsymbol{\eta}}(\mathbf{Y} \mid \boldsymbol{\theta})$, as demonstrated in **Experiments 2**, **4**, and **5**. If the variance in Eq. 9 is zero, the estimated marginal likelihood is constant across the parameter space Θ. In other words, the approximation is self-consistent (cf. Figure 1). The following proposition warrants the functional equivalence between using the variance in Eq. 8 and a tractable version of the more stable formulation in Eq. 9. The proof of the proposition is given in **Appendix B**.

Proposition 1. *Let $\pi(\boldsymbol{\theta})$ be any proposal distribution with the same support as $p(\boldsymbol{\theta} \mid \mathbf{Y})$, $\mathbf{Y}$ be a fixed data set, and f be any monotonic function, then*

$$\mathrm{Var}_{\pi(\boldsymbol{\theta})} \left(f \left(\frac{p(\mathbf{Y} \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta})}{q(\boldsymbol{\theta} \mid \mathbf{Y})} \right) \right) = 0 \implies$$
$$\mathrm{Var}_{p(\boldsymbol{\theta} \mid \mathbf{Y})} \left(\frac{p(\mathbf{Y} \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta})}{q(\boldsymbol{\theta} \mid \mathbf{Y})} \right) = 0.$$

This proposition states that (i) minimizing the variance of the log marginal likelihood is equivalent to targeting the correct quantity in Eq. 8; and (ii) we can take a different proposal than the unknown posterior $p(\boldsymbol{\theta} \mid \mathbf{Y})$ and still minimize the correct variance term in Eq. 8. For example, taking the logarithm of the marginal likelihood is akin to using the Gibbs loss in favor of the marginal likelihood for measuring predictive performance (Watanabe, 2009).

However, using a different proposal distribution may significantly change the empirical behavior of the Monte Carlo estimate and exhibit poor pre-asymptotic properties, especially if the proposal $\pi(\boldsymbol{\theta})$ has much larger variance than the true posterior $p(\boldsymbol{\theta} \mid \mathbf{Y})$. Thus, Section 3.3 discusses techniques for mitigating such behavior by using optimization schedules. Finally, the same proposition can be shown to hold when both the posterior and the likelihood in the marginal likelihood computation are replaced with neural surrogates. As a consequence, we would expect that the (reducible) variance in the doubly approximate marginal likelihood will be larger; still our experiments show measurable benefits even when using an approximate likelihood for estimating self-consistency (see **Experiments 2, 3**, and **5**).

3.3. Monte Carlo Estimation

The self-consistency loss $\mathcal{L}_{\mathrm{SC}}$ can be seamlessly added to NPE or NPLE losses. For instance, using the maximum likelihood loss for NPE with normalizing flows, we obtain:

$$\mathcal{L}_{\text{SC-NPE}}(\boldsymbol{\phi}) = \mathbb{E}_{p(\mathbf{Y})} \Bigg[\underbrace{\mathbb{E}_{p(\boldsymbol{\theta} \mid \mathbf{Y})} \big[-\log q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \mathbf{Y}) \big]}_{\text{NPE loss (on fixed } \mathbf{Y})} + \underbrace{\lambda\, \mathrm{Var}_{\pi(\boldsymbol{\theta})} \left(\log \frac{p(\boldsymbol{\theta})\, p(\mathbf{Y} \mid \boldsymbol{\theta})}{q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \mathbf{Y})} \right)}_{\text{self-consistency loss } \mathcal{L}_{\mathrm{SC}} \text{ with weight } \lambda \geq 0} \Bigg] \tag{10}$$

The variance in Eq. 9 must be empirically estimated based on finite samples $\{\boldsymbol{\theta}_k\}_{k=1}^K$ from some proposal distribution $\pi(\boldsymbol{\theta})$, as detailed in Algorithm 1. Due to the probabilistic symmetry of the joint distribution, high-density regions in the approximate posterior have the potential to cause large deviations in the estimated marginal likelihood landscape (cf. Figure 1). Consequently, we choose the approximate posterior as the proposal, $\pi(\theta) := q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \,|\, \mathbf{Y})$, to render the Monte Carlo estimate efficient.

Using the approximate posterior as a proposal has one caveat: *It is spectacularly bad at very early stages of training*. To mitigate this, we use a schedule $\mu(\cdot)$ that anneals the weight λ during training. The use of progressive annealing has been established as a successful means to control the influence of a potentially unstable approximation loop (e.g., for consistency models, Song et al., 2023; Song & Dhariwal, 2023; Schmitt et al., 2023a). Based on our experiments, we recommend choosing the schedule so that the self-consistency loss is inactive at the start of training, $\mu(0) = 0$, and its weight λ increases as training progresses.

3.4. Intuition for Benefits of Self-Consistency

So far, we presented the theoretical foundation of our self-consistency loss, and Section 5 will empirically demonstrate its effectiveness. However, one pivotal question remains: *Why* does the self-consistency loss improve inference?

In a nutshell, the self-consistency loss leverages more information for correctly amortizing $p(\boldsymbol{\theta} \,|\, \mathbf{Y})$ than isolated NPE or NLE *with the same number of simulated data sets in the training phase* through two strategies. First, the self-consistency loss informs the learned posterior and likelihood about their relation to each other, and explicitly rewards correct marginal likelihood estimates in addition to the maximum likelihood training objectives of NPE and NLE. This substantially reduces the space of admissible solutions in the training objective, upon which the correct solution is found via the maximum likelihood loss. Second, the self-consistency loss uses analytical density information from the Bayesian joint model (i.e., likelihood and prior). Since the maximum likelihood losses in NPE and NLE already optimize the networks towards correct sampling based on simulator outputs, the self-consistency loss further enhances the effect by penalizing deviations in the approximate posterior density (and approximate likelihood, if learned) in regions not immediately covered by the simulator.

4. Related Work

The self-consistency property in Eq. 4 can be used to compute importance sampling weights during inference to reweigh the approximate posterior samples (Dax et al., 2023; Glöckler et al., 2022). Importance sampling has been shown to improve the quality of the approximate posteriors when high accuracy is desired. However, it also increases the necessary amount of computations *during inference*, which in turn impedes low-latency tasks. In contrast, our self-consistent approximators do not have an increased sampling time during inference. Similarly, Glöckler et al. (2022) propose a variational approach to sequential (non-amortized) SBI which entails self-normalized importance sampling based on the marginal likelihood.

Radev et al. (2023) use a tandem of two normalizing flows for posterior and likelihood estimation that enables amortized marginal likelihood estimation during *inference*. Our self-consistency loss additionally applies the principle of amortized marginal likelihood estimation to the neural network *training* phase: By evaluating the self-consistency of the networks' (log) marginal likelihood estimates, we supply an additional training signal and thereby use the available data more efficiently. From a different perspective, our self-consistent method with learned likelihoods (SC-NPLE) can be seen as an improvement to jointly amortized neural approximation, which trains the neural approximators in isolation. In contrast, our self-consistency loss connects the networks during training and explicitly rewards accurate marginal likelihood estimation by the neural network *tandem*. As such, our work is among only a few approaches that explicitly consider the connection between different quantities in SBI and ABI (for other examples, see Brehmer et al., 2020b; Chen et al., 2023; Schmitt et al., 2023b).

A recent review of change-of-variable formulas in generative modeling discusses a multitude of self-consistency properties in (Bayesian) generative models (Köthe, 2023). Our self-consistency property of the marginal likelihood in Eq. 4 is one out of multiple possible self-consistency requirements. The work by Köthe (2023) complements the theoretical foundation of our proposed self-consistency loss, which aims to transform their theoretical remarks into a set of actionable methods for amortized Bayesian inference.

5. Empirical Evaluation

We evaluate our self-consistent estimator across a range of synthetic tasks and real-world problems. Our main baseline is NPE for tasks with an explicit likelihood, and NPLE for tasks with an implicit likelihood that is learned in tandem.

The key evaluation metric is the maximum mean discrepancy (MMD; Gretton et al., 2012) between true and approximate posterior samples. In addition, we use simulation-based calibration (SBC; Talts et al., 2018; Säilynoja et al., 2022) to assess the approximators' uncertainty quantification. Given a true posterior distribution $p(\boldsymbol{\theta} \,|\, \mathbf{Y})$, all inter-

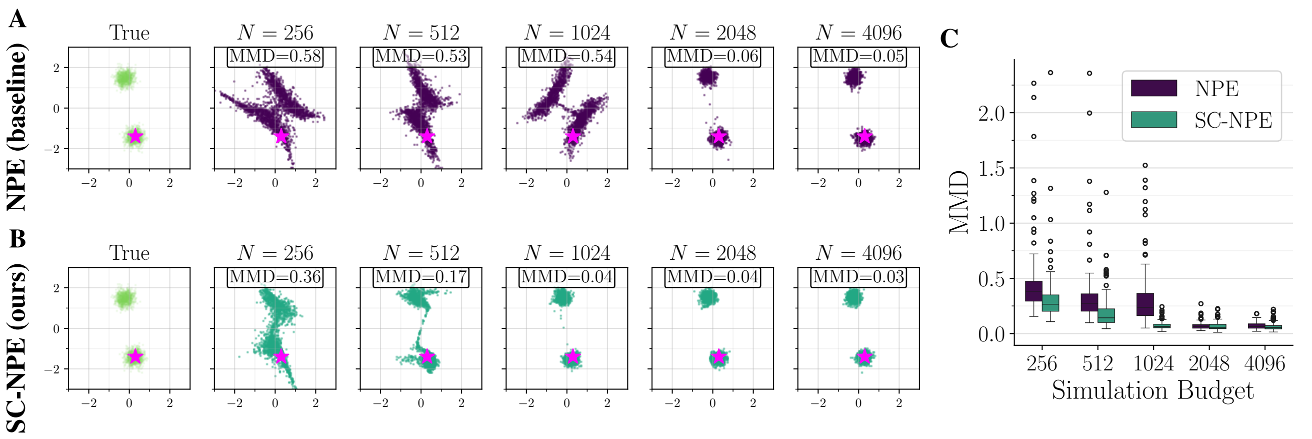


Figure 2: **Experiment 1 (Gaussian Mixture Model)**. Performance comparison between the NPE baseline (**A**) and our self-consistent SC-NPE method (**B**). Pink star ★ marks the ground-truth parameter $\boldsymbol{\theta}^*$. Both qualitative assessments (sampling) and quantitative measures (MMD; lower is better) indicate that the our SC-NPE method yields significantly better results given the same neural architecture and training budget. Across all simulation budgets, our self-consistent approximator outperforms the NPE baseline, as indexed by improved posterior fidelity (lower MMD) on 100 unseen test instances **(C)**.

vals $U_q(\theta \mid \mathbf{Y})$ are calibrated for every quantile $q \in (0, 1)$,

$$q = \iint \mathbf{I}[\theta_* \in U_q(\boldsymbol{\theta} \mid \mathbf{Y})]\, p(\mathbf{Y} \mid \boldsymbol{\theta}_*)\, p(\boldsymbol{\theta}_*) \mathrm{d}\boldsymbol{\theta}_* \mathrm{d}\mathbf{Y}, \quad (11)$$

with indicator function $\mathbf{I}[\cdot]$ (Bürkner et al., 2023). An approximate posterior may violate this equation, resulting in insufficient calibration.

5.1. Experiment 1: Gaussian Mixture Model

We first illustrate our method on a 2-dimensional Gaussian mixture model as described in Geffner et al. (2023). The model consists of two symmetrical, equally weighted components with a shared, known covariance matrix. A simulated dataset contains ten independent and identically distributed observations $\mathbf{Y} = \{\mathbf{y}_j\}_{j=1}^{10}$, generated by first sampling a parameter $\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\theta} \mid \mathbf{0}, \mathbf{I})$, and then conditionally sampling each observation as

$$\mathbf{y}_j \mid \boldsymbol{\theta} \sim 0.5\, \mathcal{N}(\mathbf{y} \mid \theta, \mathbf{I}/2) + 0.5\, \mathcal{N}(\mathbf{y} \mid -\boldsymbol{\theta}, \mathbf{I}/2). \quad (12)$$

We investigate the effect of simulation budget variations on performance, maintaining a fixed self-consistency sample size of $K = 10$ and scaling the simulation budget between $N = 256$ and $N = 4096$, each time doubling the previous budget. Both NPE (baseline) and SC-NPE (ours) train an identical neural spline flow architecture (Durkan et al., 2019) for 35 epochs. We choose a stepwise constant annealing schedule for the self-consistency weight λ such that $\lambda = 0$ for the first 5 epochs, and $\lambda = 1$ for the remaining 30 epochs. **Appendix C** contains further training details.

Results. Our method demonstrates clear superiority over the baseline, particularly at lower budget levels, as detailed in the following. Figure 2**A** and 2**B** give a visual illustration of the posterior distributions for a single dataset. Figure 2**C** shows the distribution of MMDs, computed over 100 test datasets for our self-consistent method against the NPE baseline. Qualitatively, the samples generated by SC-NPE (ours) are visually closer to the target distribution. Quantitatively, our self-consistent method achieves substantially lower MMD scores than the NPE baseline, indicating a more accurate approximation. This underscores the efficiency of integrating self-consistency when learning amortized posteriors, highlighting how our approach improves inference performance with limited computational resources. All approximators are well-calibrated (see **Appendix C.1**).

Ablation: Number of Monte Carlo Samples. We vary the number $K \in \{10, 100, 500\}$ of Monte Carlo samples to estimate the variance in the self-consistency loss with NPE (2) in **Appendix C.2**. While we observe satisfactory calibration for all self-consistent architectures, increasing the number of consistency samples beyond $K = 10$ does not noticeably improve performance in this experiment.

Variation: Approximate Neural Likelihood. We parallel **Experiment 1** with an approximate likelihood and a simulation budget of $N = 1024$ in **Appendix C.3**. Once again, our self-consistent approximator shows superior performance with respect to density estimation and sampling.

Extension: Sequential NPE with Self-Consistency. We observe that sequential neural posterior estimation (SNPE; Greenberg et al., 2019) also benefits from adding our self-consistency loss during training, as evidenced by more accurate posterior samples (see **Appendix C.4**). This result further underscores the modularity and flexibility of our proposed self-consistency loss.

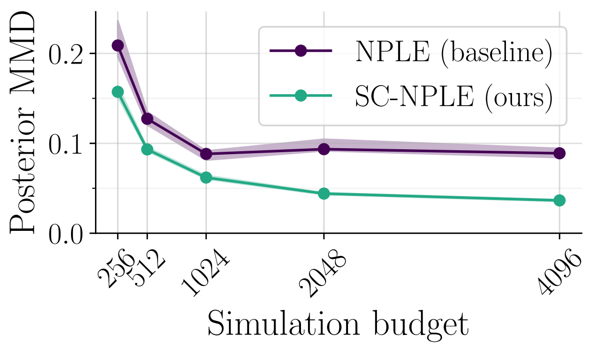


Figure 3: **Experiment 2 (Two Moons).** Our self-consistency loss yields a lower posterior error (MMD) than the baseline NPLE algorithm on the test set with equal architecture. We repeat the experiment 5 times on the same training set; plots show the median, best, and worst run.

5.2. Experiment 2: Two Moons

The two moons benchmark is characterized by a bimodal posterior with two crescents, which a posterior approximator needs to recover (Greenberg et al., 2019; Lueckmann et al., 2021; Wiqvist et al., 2021; Radev et al., 2023; Schmitt et al., 2023a). We simultaneously learn an approximate posterior and a surrogate likelihood (i.e., NPLE). Hence, the self-consistency loss will be completely simulation-based and use the learned surrogate instead of an explicit likelihood (cf. Algorithm 1, case II). We repeat the experiment for different training budgets $M \in \{256, 512, 1024, 2048, 4096\}$ to assess the performance under varying data availability. While $M = 256$ is a very small budget for the two moons benchmark, $M = 4192$ is generally considered sufficient for this experiment. We fix the number of Monte Carlo samples to estimate the variance in Eq. 9 to $K = 10$ and repeat the training loop five times for each architecture to gauge the stochasticity of the training. We evaluate the approximators' ability to estimate (i) the posterior; (ii) the likelihood; and (iii) the marginal likelihood. The **Appendix** contains all neural network training details.

Results. Our self-consistent approximator SC-NPLE consistently outperforms the baseline NPLE algorithm with respect to posterior estimation across all simulation budgets, as indexed by a better (lower) average posterior MMD on 100 unseen test instances across 5 training repetitions (see Figure 3). In this experiment, SC-NPLE (ours) only needs a simulation budget of $M = 512$ to perform on-par with the NPLE baseline that was trained on $8\times$ the simulation budget (i.e., $M = 4096$). Further, we observe a more stable training for SC-NPLE: The best and worst training runs in Figure 3 have almost equal performance, while the posterior accuracy of NPLE varies between repetitions. Table 1 shows the estimated likelihood density of an observed data set $\mathbf{Y}_{\text{real}}$ given the ground-truth parameter $\boldsymbol{\theta}^*$ which was used to simulate the data set. While the credible intervals of NPLE (baseline) and SC-NPLE (ours) across the test set have substantial overlap and perform on-par for small simulation budgets, our self-consistent approximator assigns higher (better) likelihood densities to the ground-truth at large simulation budgets. It is worth noting that the width of the CI is smaller for SC-NPLE, which is a desirable property indicating a reduced approximation error. Finally, we estimate the marginal likelihood of 500 unseen test instances and observe that the estimates from our self-consistent approximator are significantly sharper (smaller width of the 95% for fixed data sets), which is an indicator for a reduced approximation error (see Table 2).

Table 1: **Experiment 2 (Two Moons).** Log likelihood density of the observed data $\mathbf{Y}_{\text{real}}$ under the true data-generating parameter $\boldsymbol{\theta}^*$ (higher is better). We report the mean±SE across 1000 unseen test instances.

Method	N=512	N=1024	N=2048	N=4096
NPLE	**3.15**±0.03	3.18±0.03	2.88±0.04	2.91±0.05
SC-NPLE	3.14±0.02	**3.45**±0.02	**3.71**±0.02	**3.90**±0.02

5.3. Experiment 3: Oscillatory Hes1 Expression Model

As a scientific real-world example, we apply our method to an experimental data set in biology (Silk et al., 2011). Upon serum stimulation of certain cell lines, the transcription factor Hes1 exhibits sustained oscillatory transcription patterns (Momiji & Monk, 2008). The measured concentration of Hes1 mRNA is modeled with a set of three differential equations which are governed by four parameters $\boldsymbol{\theta} = (p_0, h, k_1, \nu)$ with fixed initial conditions according to Filippi et al. (2011), see **Appendix E** for details.

We use a fixed simulation budget of $N = 512$ data sets for neural network training. This simulation budget enables amortized inference, yet it is orders of magnitude smaller than the required budget for approximate Bayesian computation algorithms (e.g., ABC-SMC; Sisson et al., 2007) for

Table 2: **Experiment 2 (Two Moons).** Approximation error of the log marginal likelihood (LML) estimate (lower is better). Our self-consistent estimator yields a significantly smaller approximation error, as indicated by sharper LML estimates. For a data set $\mathbf{Y}_{\text{real}}$, the approximation error is quantified as the width of the LML estimate's 95% CI. We report its mean±SE across 1000 unseen test instances.

Method	N=512	N=1024	N=2048	N=4096
NPLE	6.51±0.11	7.28±0.10	9.07±0.06	10.21±0.08
SC-NPLE	**1.70**±0.02	**1.37**±0.02	**1.21**±0.01	**1.14**±0.01

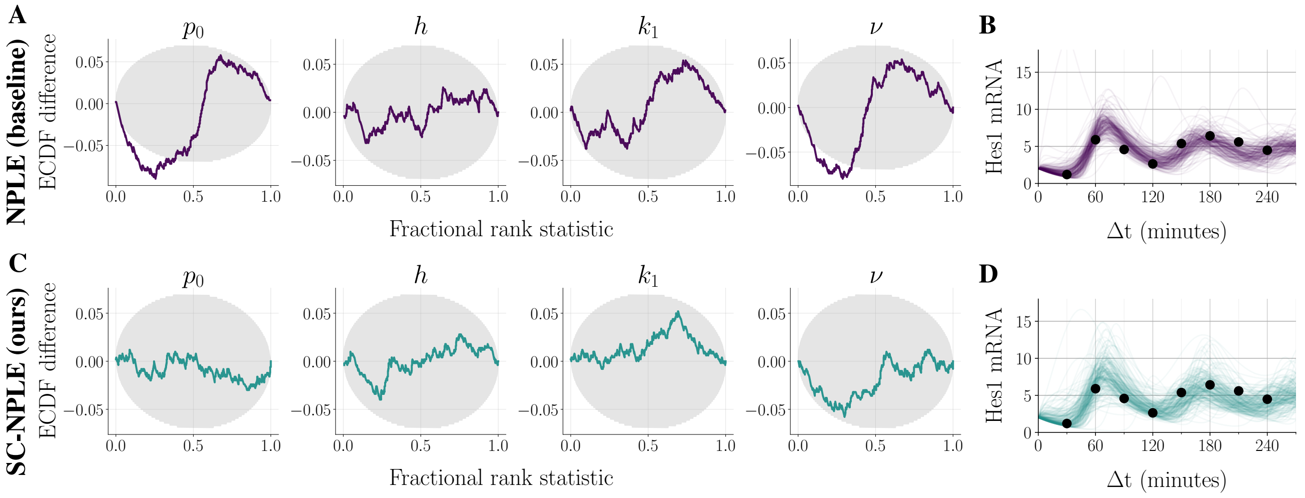


Figure 4: **Experiment 3 (Hes1 Expression).** The baseline NPLE approximator shows deficient simulation-based calibration, as indexed by ECDF lines outside the gray 95% confidence bands **(A)**. In contrast, our self-consistent approximator is well-calibrated **(C)**. Samples from the posterior predictive distribution on real experimental data (black dots; Silk et al., 2011) are comparable between NPLE **(B)** and SC **(D)**.

a single observed data set (Silk et al., 2011). We train a neural surrogate likelihood in tandem with the neural posterior (NPLE) to use an approximate likelihood $q_{\boldsymbol{\eta}}(\mathbf{Y} \mid \boldsymbol{\theta})$ in the self-consistency loss. The self-consistent approximator uses $K = 500$ Monte Carlo samples. The annealing schedule yields $\lambda = 0$ for the first 10 epochs, and $\lambda = 1$ for the remaining 60 epochs. See **Appendix E** for details.

Results. Our self-consistent approximator with approximate likelihood shows superior simulation-based calibration compared to the NPLE baseline, particularly with respect to the parameters p_0 and ν (see Figure 4**A** and **C**). The posterior predictive distributions of both methods have a similar fit to the real experimental time series $\mathbf{Y}_{\text{real}}$ from Silk et al. (2011), see Figure 4**B** and 4**D**.

Ablation: Underexpressive likelihood network. For models where the likelihood is only implicitly defined, an auxiliary likelihood surrogate must be learned. One possible concern is that enforcing self-consistency between the surrogate likelihood and the posterior might actually hurt posterior inference if the likelihood is much more challenging to approximate than the posterior. We repeat **Experiment 3** with an underexpressive likelihood network that only features linear units. By ensuring that the likelihood surrogate is insufficient by design, we can emulate situations in which the approximation of the likelihood is unreliable and observe its impact on posterior inference. We observe no substantial drop in posterior performance compared to the reference posterior (see **Appendix** E for details).

5.4. Experiment 4: Source Location Finding

We adapt the source finding experiment from Foster et al. (2021), which involves finding the location $\boldsymbol{\theta}$ of a hidden source in 2D. The source emits a signal whose intensity decays inversely with the square of distance. We observe a noise corrupted version $\mathbf{Y}$ of that signal through $N = 30$ fixed measurement points. We systematically vary the number of Monte Carlo samples to estimate the variance in Eq. 9 as $K \in \{5, 10, 20, 50, 100, 500\}$. Both NPE (baseline) and SC-NPE (ours) use identical neural networks and are trained for 35 epochs on identical settings to ensure a fair comparison. The annealing schedule for the self-consistency loss weight λ is piecewise constant: It yields λ for the first 20% of the training loop, then switching to

Table 3: **Experiment 4 (Source Location Finding).** MMD as a function of the simulation budget N and the number K of Monte Carlo samples in Eq. 9. We report mean±SE across 1000 unseen test instances (lower is better). Our self-consistent approximators outperform the NPE baseline throughout all simulation budgets, and the advantage is particularly attenuated at low budgets.

		$N = 512$	$N = 1024$	$N = 2048$	$N = 4096$
NPE		0.54 ± 0.02	0.39 ± 0.02	0.25 ± 0.02	0.22 ± 0.01
SC-NPE (Ours)	K=5	0.47 ± 0.03	0.29 ± 0.02	$\mathbf{0.24} \pm 0.02$	$\mathbf{0.20} \pm 0.02$
	K=10	0.53 ± 0.04	$\mathbf{0.27} \pm 0.02$	$\mathbf{0.24} \pm 0.02$	0.22 ± 0.02
	K=20	0.51 ± 0.03	0.32 ± 0.02	$\mathbf{0.24} \pm 0.02$	0.23 ± 0.02
	K=50	0.52 ± 0.04	0.30 ± 0.02	0.26 ± 0.02	0.21 ± 0.02
	K=100	0.42 ± 0.03	0.29 ± 0.02	0.25 ± 0.02	0.21 ± 0.02
	K=500	$\mathbf{0.40} \pm 0.03$	0.32 ± 0.02	0.25 ± 0.02	0.21 ± 0.02

Table 4: **Experiment 5 (High-dimensional time series model).** MMD as a function of the number K of self-consistency Monte Carlo samples and a simulation budget $N = 2048$. We report mean±SE across 50 test instances.

NPLE	SC-NPLE (Ours)			
	$K = 5$	$K = 10$	$K = 50$	$K = 100$
0.88 ± 0.06	0.78 ± 0.04	0.75 ± 0.05	0.69 ± 0.04	**0.54** ± 0.03

$\lambda = 0.01$. **Appendix** F contains details on the neural network architectures and training scheme.

Results. Table 3 reports the MMD between amortized posterior approximation and a reference posterior from Hamiltonian Monte Carlo (HMC; as implemented in Stan, Carpenter et al., 2017). Our self-consistent method demonstrates superior performance to the baseline NPE across all simulation budgets and number of SC samples K. The performance advantage is particularly pronounced at lower simulation budgets. Increasing the number of SC samples does not generally result in significantly improved performance.

5.5. Experiment 5: High-Dimensional Time Series Model

We demonstrate the effectiveness of the self-consistency loss for high-dimensional data without assuming an explicit likelihood. To this end, we implement an autoregressive compartmental time series model where the data $\mathbf{Y}$ is a 160-dimensional vector. We simultaneously learn the posterior and likelihood (NPLE) based on $N = 2048$ training examples. The task of learning the likelihood is directly affected by the high data dimensionality since we learn the likelihood in the uncompressed 160-dimensional data space. We benchmark standard NPLE against NPLE including our self-consistency loss (SC-NPLE) with $K \in \{5, 10, 50, 100\}$ Monte Carlo samples to estimate the variance in Eq. 9.

Results. We report the maximum mean discrepancy (MMD) to a HMC reference posterior across 50 unseen test instances. Table 4 shows the results, demonstrating a clear trend of monotonic performance improvement with an increasing number K of self-consistency Monte Carlo samples. This demonstrates the effectiveness of our self-consistency loss, even when dealing with high-dimensional data where the likelihood estimation can be notably more challenging.

6. Conclusion

We proposed a new method to exploit inherent symmetry in a joint probabilistic model $p(\boldsymbol{\theta}, \mathbf{Y})$ to improve amortized Bayesian inference. Across four experiments, we illustrated that the combination of simulation-based inference and (approximate) likelihood-based learning increases the efficiency of neural posterior and likelihood estimation. Concretely, we demonstrated that an additional self-consistency loss leads to (i) better posterior densities; (ii) better posterior samples; (iii) better likelihood densities; and (iv) sharper marginal likelihood estimates. The advantage of our self-consistent estimator is particularly evident for low data scenarios, which is a frequent bottleneck in real-world applications of simulation-based inference (e.g., Zhang & Mikelsons, 2023; Zeng et al., 2023; Bharti et al., 2022)

Limitations. As always, there is no free lunch: The improved performance though our self-consistency loss comes with an increased computational cost. However, the self-consistency loss is designed to pre-pay the cost *during the training stage*. As a consequence, the inference algorithm remains unaltered and we maintain rapid amortized inference. Hence, the trade-off is ideal for applied scenarios where the upfront training time is not a bottleneck, but training data is scarce and fast inference is desired. Further, our method relies on the ability to evaluate the prior density $p(\boldsymbol{\theta})$. This currently limits its applicability to scenarios where the prior density is available in analytic form (most probabilistic modeling applications) or can be learned.

Outlook. While this paper focused on amortized Bayesian inference with normalizing flows, our self-consistency loss can readily be applied to sequential simulation-based inference (Papamakarios et al., 2019; Greenberg et al., 2019; Glöckler et al., 2022; Wiqvist et al., 2021). Likewise, other conditional density estimators like score modeling (Geffner et al., 2023; Sharrock et al., 2022; Pacchiardi & Dutta, 2022), flow-matching (Lipman et al., 2023), or consistency models (Schmitt et al., 2023a) may benefit from our additional loss function as well. Finally, future research could explore variations and extensions of our proposed method, such as different proposal distributions $\pi(\boldsymbol{\theta})$ for more efficient Monte Carlo estimates, prior density learning, likelihood learning on summaries instead of the raw data, or improved loss functions altogether that build on the principle of self-consistency. **Appendix A** contains a selection of Frequently Asked Questions (FAQ) that a reader might have.

Code Availability

We provide reproducible code in the open repository at `https://github.com/marvinschmitt/self-consistency-abi`

Impact Statement

This paper presents work that advances the field of amortized Bayesian inference (ABI) by rendering analyses more data-efficient. As such, our method can be used to improve the results in malign applications as well, and its societal implications need to be evaluated on an individual basis.

Acknowledgments

MS was supported by the Cyber Valley Research Fund (grant number: CyVy-RF-2021-16) and the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy EXC-2075 - 390740016 (the Stuttgart Cluster of Excellence SimTech). DRI is supported by EPSRC through the Modern Statistics and Statistical Machine Learning (StatML) CDT programme, grant no. EP/S023151/1. DH was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under project 508399956. UK was supported by the Informatics for Life initiative funded by the Klaus Tschira Foundation.

References

Alexanderson, S. and Henter, G. E. Robust model training and generalisation with studentising flows, 2020. 14, 15, 17

Avecilla, G., Chuong, J. N., Li, F., Sherlock, G., Gresham, D., and Ram, Y. Neural networks enable efficient and accurate simulation-based inference of evolutionary parameters from adaptation dynamics. *PLoS Biology*, 20 (5):e3001633, 2022. 1, 2

Bharti, A., Filstroff, L., and Kaski, S. Approximate Bayesian computation with domain expert in the loop. In *International Conference on Machine Learning*, pp. 1893–1905. PMLR, 2022. 9

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017. 1

Brehmer, J., Cranmer, K., Louppe, G., and Pavez, J. A guide to constraining effective field theories with machine learning. *Physical Review D*, 98(5):052004, 2018. 1

Brehmer, J., Kling, F., Espejo, I., and Cranmer, K. Madminer: Machine learning-based inference for particle physics. *Computing and Software for Big Science*, 4: 1–25, 2020a. 1

Brehmer, J., Louppe, G., Pavez, J., and Cranmer, K. Mining gold from implicit models to improve likelihood-free inference. *Proceedings of the National Academy of Sciences*, 117(10):5242–5249, 2020b. 1, 5

Bürkner, P.-C., Scholz, M., and Radev, S. T. Some models are useful, but how do we know which ones? towards a unified bayesian model taxonomy. *Statistics Surveys*, 17 (none), 2023. ISSN 1935-7516. doi: 10.1214/23-ss145. 6

Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., and Riddell, A. Stan: A probabilistic programming language. *Journal of statistical software*, 76(1), 2017. 1, 9

Chen, Y., Gutmann, M. U., and Weller, A. Is learning summary statistics necessary for likelihood-free inference? In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 4529–4544. PMLR, 2023. 5

Cranmer, K., Brehmer, J., and Louppe, G. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 2020. 1

Dax, M., Green, S. R., Gair, J., Pürrer, M., Wildberger, J., Macke, J. H., Buonanno, A., and Schölkopf, B. Neural importance sampling for rapid and reliable gravitational-wave inference. *Physical Review Letters*, 130(17), April 2023. ISSN 1079-7114. doi: 10.1103/physrevlett.130. 171403. 5

Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G. Neural spline flows. *Advances in neural information processing systems*, 32, 2019. 6, 17, 18, 19

Filippi, S., Barnes, C., Cornebise, J., and Stumpf, M. P. H. On optimality of kernels for approximate bayesian computation using sequential monte carlo, June 2011. 7, 17

Foster, A., Ivanova, D. R., Malik, I., and Rainforth, T. Deep adaptive design: Amortizing sequential bayesian experimental design. In *International Conference on Machine Learning*, 2021. 8

Geffner, T., Papamakarios, G., and Mnih, A. Compositional score modeling for simulation-based inference. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 11098–11116. PMLR, 23–29 Jul 2023. 2, 6, 9

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. *Bayesian Data Analysis (3rd Edition)*. Chapman and Hall/CRC, 2013. 1

Glöckler, M., Deistler, M., and Macke, J. H. Variational methods for simulation-based inference. In *International Conference on Learning Representations*, 2022. 3, 5, 9

Gonçalves, P. J., Lueckmann, J.-M., Deistler, M., et al. Training deep neural density estimators to identify mechanistic models of neural dynamics. *Elife*, 2020. 1, 2

Greenberg, D., Nonnenmacher, M., and Macke, J. Automatic posterior transformation for likelihood-free inference. In *International Conference on Machine Learning*, 2019. 6, 7, 9, 13, 16

Gretton, A., Borgwardt, K., Rasch, M., Schölkopf, B., and Smola, A. A Kernel Two-Sample Test. *The Journal of Machine Learning Research*, 13:723–773, 2012. 5

Hüllermeier, E. and Waegeman, W. Aleatoric and Epistemic Uncertainty in Machine Learning: An Introduction to Concepts and Methods. *Machine Learning*, 110(3):457–506, March 2021. ISSN 0885-6125, 1573-0565. doi: 10.1007/s10994-021-05946-3. 1, 13

Köthe, U. A review of change of variable formulas for generative modeling, 2023. 4, 5

Lavin, A., Zenil, H., Paige, B., et al. Simulation intelligence: Towards a new generation of scientific methods. *arXiv preprint*, 2021. 1

Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., and Teh, Y. W. Set transformer: A framework for attention-based permutation-invariant neural networks. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3744–3753. PMLR, 2019. 13, 18

Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., and Le, M. Flow matching for generative modeling. In *The 11th International Conference on Learning Representations*, 2023. 9

Lueckmann, J.-M., Boelts, J., Greenberg, D., Goncalves, P., and Macke, J. Benchmarking simulation-based inference. In Banerjee, A. and Fukumizu, K. (eds.), *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 343–351. PMLR, 13–15 Apr 2021. 7, 17, 19

Meng, X.-L. and Wong, W. H. Simulating ratios of normalizing constants via a simple identity: a theoretical exploration. *Statistica Sinica*, 1996. 3

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018. 17

Momiji, H. and Monk, N. A. Dissecting the dynamics of the hes1 genetic oscillator. *Journal of Theoretical Biology*, 254(4):784–798, oct 2008. doi: 10.1016/j.jtbi.2008.07.013. 7, 17

Neal, R. M. *MCMC using Hamiltonian dynamics*. May 2011. doi: 10.1201/b10905. 1

Pacchiardi, L. and Dutta, R. Score matched neural exponential families for likelihood-free inference. *J. Mach. Learn. Res.*, 23:38–1, 2022. 9

Papamakarios, G., Sterratt, D., and Murray, I. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 837–848. PMLR, 2019. 9

Radev, S. T., Mertens, U. K., Voss, A., Ardizzone, L., and Köthe, U. BayesFlow: Learning complex stochastic models with invertible neural networks. *IEEE transactions on neural networks and learning systems*, 2020. 1, 2, 13, 14, 18

Radev, S. T., Schmitt, M., Pratz, V., Picchini, U., Köthe, U., and Bürkner, P.-C. JANA: Jointly Amortized Neural Approximation of Complex Bayesian Models. In Evans, R. J. and Shpitser, I. (eds.), *Proceedings of the 39th Conference on Uncertainty in Artificial Intelligence*, volume 216 of *Proceedings of Machine Learning Research*, pp. 1695–1706. PMLR, 2023. 3, 5, 7, 13

Rezende, D. J. and Mohamed, S. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pp. 1530–1538. JMLR.org, 2015. 2

Säilynoja, T., Bürkner, P.-C., and Vehtari, A. Graphical test for discrete uniformity and its applications in goodness-of-fit evaluation and multiple sample comparison. *Statistics and Computing*, 32(2):1–21, 2022. 5, 15

Schmitt, M., Pratz, V., Köthe, U., Bürkner, P.-C., and Radev, S. T. Consistency models for scalable and fast simulation-based inference, 2023a. 2, 5, 7, 9

Schmitt, M., Radev, S. T., and Bürkner, P.-C. Fuse it or lose it: Deep fusion for multimodal simulation-based inference, 2023b. 5

Sharrock, L., Simons, J., Liu, S., and Beaumont, M. Sequential neural score estimation: Likelihood-free inference with conditional score based diffusion models, 2022. 2, 9

Silk, D., Kirk, P. D., Barnes, C. P., Toni, T., Rose, A., Moon, S., Dallman, M. J., and Stumpf, M. P. Designing attractive models via automated identification of chaotic and oscillatory dynamical regimes. *Nature Communications*, 2(1), 2011. doi: 10.1038/ncomms1496. 7, 8, 17

Sisson, S. A., Fan, Y., and Tanaka, M. M. Sequential monte carlo without likelihoods. *Proceedings of the National*

Academy of Sciences, 104(6):1760–1765, February 2007. doi: 10.1073/pnas.0607208104. 7

Song, Y. and Dhariwal, P. Improved Techniques for Training Consistency Models, October 2023. 5

Song, Y., Dhariwal, P., Chen, M., and Sutskever, I. Consistency models. In *International Conference on Machine Learning*, 2023. 5

Talts, S., Betancourt, M., Simpson, D., Vehtari, A., and Gelman, A. Validating Bayesian inference algorithms with simulation-based calibration. *arXiv preprint*, 2018. 5, 15

Vehtari, A., Gabry, J., Magnusson, M., Yao, Y., Bürkner, P.-C., Paananen, T., and Gelman, A. loo: Efficient leave-one-out cross-validation and WAIC for Bayesian models, 2022. 3

Watanabe, S. *Algebraic geometry and statistical learning theory*, volume 25. Cambridge university press, 2009. 4

Wiqvist, S., Frellsen, J., and Picchini, U. Sequential neural posterior and likelihood approximation. *arXiv preprint*, 2021. 3, 7, 9

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R., and Smola, A. Deep sets, 2017. 13, 14, 15

Zeng, J., Todd, M. D., and Hu, Z. Probabilistic damage detection using a new likelihood-free Bayesian inference method. *Journal of Civil Structural Health Monitoring*, 13(2-3):319–341, 2023. 9

Zhang, Y. and Mikelsons, L. Sensitivity-guided iterative parameter identification and data generation with BayesFlow and PELS-VAE for model calibration. *Advanced Modeling and Simulation in Engineering Sciences*, 10(1):1–28, 2023. 9

APPENDIX

A. Frequently Asked Questions (FAQ)

Q: How can I reproduce the results?

The code is hosted in a repository at `https://github.com/marvinschmitt/self-consistency-abi`

Q: How does the self-consistency loss change my current amortized Bayesian workflow?

If you have access to an explicit likelihood, you need to implement a `likelihood.log_prob` method to evaluate $p(\mathbf{Y} \mid \boldsymbol{\theta})$. This is straightforward with common frameworks such as `tensorflow_probability` or `scipy.stats`. If your likelihood is implicit (fully simulation-based), your neural likelihood approximator needs to yield a tractable density (e.g., through a normalizing flow). Further, you need to implement a `prior.log_prob` method for your prior distribution, regardless of whether you use NPE (explicit likelihood) or NPLE (implicit likelihood). Alternatively, you may try to learn the prior density with an unconditional density estimator on-the-fly (see Section 6).

Q: When is it useful to add the self-consistency loss?

When the simulation program is computationally costly or the simulation budget is fixed, our experiments suggest that adding the self-consistency loss to an optimization objective might help get more out of the available training data.

Q: What about learned summary statistics?

Our method is fully compatible with end-to-end learning of summary statistics alongside the neural approximator (Radev et al., 2020; 2023). In fact, **Experiment 1** uses a DeepSet (Zaheer et al., 2017) and **Experiment 4** uses a SetTransformer (Lee et al., 2019) to learn fixed-length summary statistics $h(\mathbf{Y})$ from the observables $\mathbf{Y}$, which are then passed to the posterior approximator. As mentioned in Section 6, the likelihood

Q: When do I activate the self-consistency loss during training?

This depends on the complexity of the problem. Your approximate posterior (and approximate likelihood, if applicable) should be sufficiently good so that (i) the proposals for $\tilde{\theta}$ cover relevant regions; and (ii) the log density estimates for the posterior (and likelihood, if applicable) have an acceptable quality for the Monte Carlo estimate in Eq. 9. Further, you have freedom in designing an annealing schedule $\lambda = \mu(\cdot)$ for the weight of the self-consistency term in the loss function. For instance, you might opt for a smooth schedule which gradually increases the self-consistency weight.

Q: Why aren't the posterior samples in Experiment 1 perfectly aligned with the true parameter?

The simulated data sets in the Gaussian mixture model only consist of ten observations from the Gaussian mixture model with locations $\boldsymbol{\theta}$ and $-\boldsymbol{\theta}$. Due to aleatoric uncertainty in the data-generating process (Hüllermeier & Waegeman, 2021), the empirical information in the sample does not even suffice to inform the true posterior to concentrate on the true data-generating parameter $\boldsymbol{\theta}$. Instead, the goal of an approximate posterior is to match the true posterior including the uncertainty it encodes.

Q: Why do the NPE posterior samples look so bad in Experiment 1 at $N = 1024$? Other papers show better sampling for NPE at this budget.

We observed that NPE shows very inconsistent performance across the parameter space at low simulation budgets. In other words, NPE performs reasonably well for some parameter regions, and atrociously bad for others. In Figure 2**C**, this phenomenon manifests as a large range in the corresponding boxplot, which shows the MMD across different test instances. In contrast, our SC-NPE method achieves remarkably good performance across the entire parameter space at a simulation budget of $N = 1024$.

Q: What about non-amortized sequential algorithms, like Sequential Neural Posterior Estimation (SNPE)?

Our method can readily be integrated with sequential SBI algorithms, such as SNPE (Greenberg et al., 2019). We observed performance increases when SNPE is equipped with our self-consistency loss (see **Appendix C.4**).

B. Proof of Proposition 1

In the following, we provide a proof of Proposition 1.

Proof. First, we are going to make use of the well-known fact that the variance of a constant is zero, and zero variance implies a constant argument, that is, for any continuous random vector X, we have

$$f(X) = c \implies \mathrm{Var}(f(X)) = 0 \tag{13}$$
$$\mathrm{Var}(f(X)) = 0 \implies f(X) = c \tag{14}$$

For (13), we simply apply the definition of the variance

$$\mathrm{Var}(c) = \mathbb{E}\left[(c - \mathbb{E}\left[c\right])^2\right] \tag{15}$$
$$= \mathbb{E}\left[(c - c)^2\right] = 0, \tag{16}$$

where we used the fact that the expectation of a constant recovers the constant itself, that is, $\mathbb{E}\left[c\right] = c$.

For (14), we first note that due to the definition of the variance,

$$\mathrm{Var}(f(X)) = \mathbb{E}\left[(f(X) - \mathbb{E}\left[f(X)\right])^2\right], \tag{17}$$

the squared difference $(f(X) - \mathbb{E}\left[f(X)\right])^2$ is strictly positive, so the only way for the variance to become zero is if $(f(X) - \mathbb{E}\left[f(X)\right])^2 = 0$. This, in turn, implies that $f(X) = \mathbb{E}\left[f(X)\right]$ for any realization of X, which can only happen if $f(X) = c$.

The proof of Proposition 1 is structured in two parts. First, we show that zero variance of a monotone function directly implies zero variance for the argument. Second, we show that zero variance with respect to a distribution implies zero variance with respect to a different distribution with the same support.

For the first part, given the assumption of zero variance,

$$\mathrm{Var}_{p(\boldsymbol{\theta}\,|\,\mathbf{Y})}\left(f\left(\frac{p(\mathbf{Y}\,|\,\boldsymbol{\theta})p(\boldsymbol{\theta})}{q(\boldsymbol{\theta}\,|\,\mathbf{Y})}\right)\right) = 0, \tag{18}$$

it follows that $f(p(\mathbf{Y}\,|\,\boldsymbol{\theta})p(\boldsymbol{\theta})/q(\boldsymbol{\theta}\,|\,\mathbf{Y})) = c$, $\forall\boldsymbol{\theta} \in \Theta$. Since we assume that f is monotone, then the functional argument of $f(\cdot)$ must be constant over Θ, and so

$$\mathrm{Var}_{p(\boldsymbol{\theta}\,|\,\mathbf{Y})}\left(\frac{p(\mathbf{Y}\,|\,\boldsymbol{\theta})p(\boldsymbol{\theta})}{q(\boldsymbol{\theta}\,|\,\mathbf{Y})}\right) = 0. \tag{19}$$

For the second part, we again observe that the assumption

$$\mathrm{Var}_{\pi(\boldsymbol{\theta})}\left(f\left(\frac{p(\mathbf{Y}\,|\,\boldsymbol{\theta})p(\boldsymbol{\theta})}{q(\boldsymbol{\theta}\,|\,\mathbf{Y})}\right)\right) = 0 \tag{20}$$

implies that $f(p(\mathbf{Y}\,|\,\boldsymbol{\theta})p(\boldsymbol{\theta})/q(\boldsymbol{\theta}\,|\,\mathbf{Y})) = c$, $\forall\boldsymbol{\theta} \in \Theta$. Since, by assumption, $\pi(\boldsymbol{\theta})$ and $p(\boldsymbol{\theta}\,|\,\mathbf{Y})$ have the same support, it follows that

$$\mathrm{Var}_{p(\boldsymbol{\theta}\,|\,\mathbf{Y})}\left(f\left(\frac{p(\mathbf{Y}\,|\,\boldsymbol{\theta})p(\boldsymbol{\theta})}{q(\boldsymbol{\theta}\,|\,\mathbf{Y})}\right)\right) = 0 \tag{21}$$

Combining this with the previous result from the first part concludes the proof. □

C. Details about Experiment 1

The normalizing flow uses a heavy-tailed Student-t latent distribution with 100 degrees of freedom to provide a more robust latent space (Alexanderson & Henter, 2020). The neural spline flow has 4 coupling layers and learnable permutation layers. As a summary network, a DeepSet (Zaheer et al., 2017) learns 4-dimensional embeddings that lift the $i.i.d.$ structure of the exchangeable data $\mathbf{Y}$ and are maximally informative for posterior inference (Radev et al., 2020). The neural networks are trained for a total of 35 epochs with a batch size of 32 and an initial learning rate of 10^{-3}.

C.1. Calibration

All approximators (NPE and self-consistent ones) are well-calibrated according to simulation-based calibration (Talts et al., 2018; Säilynoja et al., 2022), as illustrated in Figure 5.

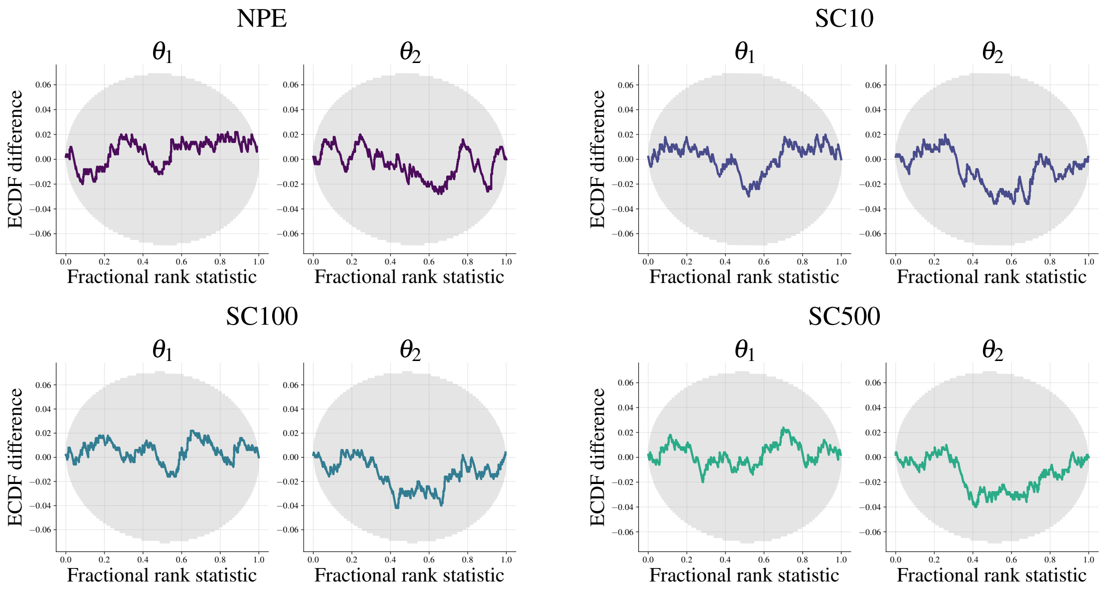


Figure 5: **Experiment 1 (Gaussian mixture model).** All approximators are well-calibrated.

C.2. Ablation: Number of Monte Carlo Samples for the Self-Consistency Loss

In this ablation, we investigate the effect of the number of Monte Carlo samples $K \in \{10, 100, 500\}$ to estimate the variance in Eq. 2. All self-consistent approximators outperform the NPE baseline with respect via (i) visually better posterior samples and density as well as (ii) a better (lower) MMD on 50 unseen test instances.

C.3. Variation: Implicit Likelihood (NPLE)

We repeat **Experiment 1** with a fully simulation-based approach which does not need an explicit likelihood to estimate the self-consistency loss. To this end, we replace the explicit likelihood $p(\mathbf{Y} \mid \boldsymbol{\theta})$ in Eq. 9 with the approximate likelihood $q_\eta(\mathbf{Y} \mid \boldsymbol{\theta})$, which is represented by a neural network and learned simultaneously with the neural posterior approximator. The full loss follows as

$$\begin{aligned}\mathcal{L}_{\text{SC-NPLE}} = \mathbb{E}_{p(\boldsymbol{\theta},\mathbf{Y})} \Big[& \underbrace{-\log q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \mathbf{Y}) - \log q_{\boldsymbol{\eta}}(\mathbf{Y} \mid \boldsymbol{\theta})}_{\text{NPLE loss}} \\ & + \underbrace{\lambda \, \mathrm{Var}_{\boldsymbol{\theta} \sim \pi(\boldsymbol{\theta})} \Big(\log p(\boldsymbol{\theta}) + \log q_{\boldsymbol{\eta}}(\mathbf{Y} \mid \boldsymbol{\theta}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \mathbf{Y}) \Big) \Big]}_{\text{self-consistency loss } \mathcal{L}_{\text{SC}} \text{ with approximate likelihood}},\end{aligned} \tag{22}$$

which is a combination of the NPLE loss and the self-consistency loss with approximate likelihood.

The simulation budget is fixed to $N = 1024$ data sets. Both NPLE and our self-consistent approximator with $K = 100$ Monte Carlo samples are trained for 35 epochs, have an identical neural spline flow architecture, have a heavy-tailed Student-t_{100} latent space (Alexanderson & Henter, 2020), and use an identical DeepSet (Zaheer et al., 2017) to learn summary statistics of the data $\mathbf{Y}$ for the posterior approximator. Since the Monte Carlo approximation in the self-consistency loss now depends on both an approximate posterior and an approximate likelihood, we only activate the self-consistency loss after 20 epochs (as opposed to 5 epochs in **Experiment 1** with an explicit likelihood).

We can benchmark the methods' performance against the true posterior since the explicit likelihood of this simulator is known (albeit inaccessible for the approximators). We confirm the results of **Experiment 1** in the fully simulation-based setting with only an implicit likelihood: Our self-consistent approximator consistently outperforms the baseline NPLE approximator with respect to posterior density and sampling (see Figure 7).

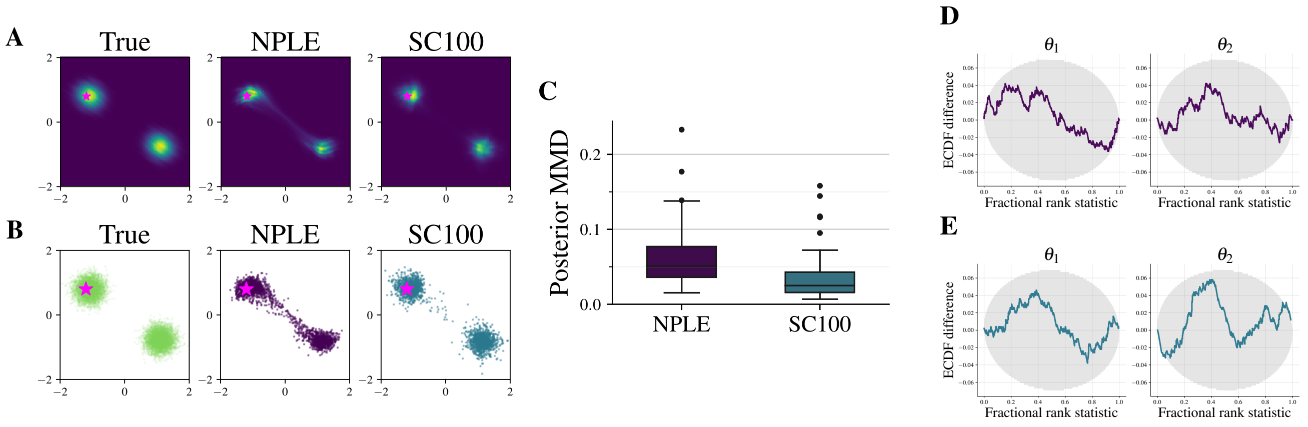


Figure 7: **Experiment 1 (Gaussian mixture model).** Our self-consistent posterior estimator (SC100) outperforms the NPLE baseline using the same neural architecture trained on an identical simulation budget. Adding the self-consistency loss leads to improved density estimation (**A**) and sampling (**B**), judged both visually and via MMD between approximate and true posteriors (**C**). Both approximators are well-calibrated **(D, E)**. Pink star ★ marks the true parameter $\boldsymbol{\theta}$.

C.4. Extension: Sequential Neural Posterior Estimation with Self-Consistency Loss

We repeat **Experiment 1** with sequential neural posterior estimation (SNPE; Greenberg et al., 2019) and observe similar improvements in the quality of posterior samples by adding our self-consistency loss (see Figure 8). This underscores the modular nature of the self-consistency loss, which renders it applicable to a variety of inference algorithms. We leave an in-depth analysis of the interplay of self-consistency losses and sequential SBI algorithms to future work, as the focus of this paper lies on *amortized* Bayesian inference.

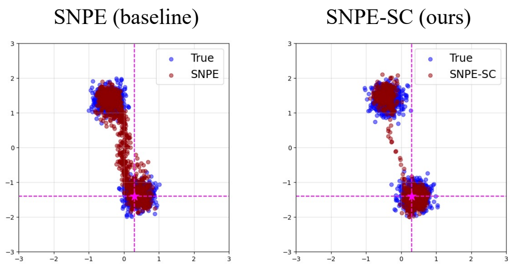


Figure 8: Our self-consistency loss visually improves posterior sampling for SNPE in **Experiment 1**.

D. Details about Experiment 2

The two moons model is described in detail elsewhere (Lueckmann et al., 2021). We use a uniform prior on both components of theta: $\theta_1, \theta_2 \sim \mathrm{Uniform}(-2, 2)$.

The neural network architectures are identical for NPLE (baseline) and SC-NPLE (ours). The posterior and likelihood networks are identical: Both consist of a neural spline flow (Durkan et al., 2019) with 6 coupling layers of 128 units each and weight regularization with a factor $\gamma = 10^{-4}$. Further, the latent space in the neural spline flow is a heavy-tailed Student-t distribution (Alexanderson & Henter, 2020) with 50 degrees of freedom. The neural networks are trained for 200 epochs with a batch size of 32 and an initial learning rate of $5 \cdot 10^{-4}$. For the self-consistent variations, we choose a piecewise constant schedule $\mu(\cdot)$ on the weight λ, where $\lambda = 0$ for the first 100 epochs (i.e., no self-consistency term) and $\lambda = 1$ for the remaining 100 epochs.

E. Details about Experiment 3

In **Experiment 3**, we apply our method to an experimental data set in biology (Silk et al., 2011). Upon serum stimulation of various cell lines, the transcription factor Hes1 exhibits sustained oscillatory transcription patterns (Momiji & Monk, 2008). The concentration of Hes1 mRNA can be modeled by a set of three differential equations,

$$\frac{\mathrm{d}m}{\mathrm{d}t} = -k_{deg}m + \frac{1}{1 + (p_2/p_0)^h}, \quad \frac{\mathrm{d}p_1}{\mathrm{d}t} = -k_{deg}p_1 + \nu m - k_1 p_1, \quad \frac{\mathrm{d}p_2}{\mathrm{d}t} = -k_{deg}p_2 + k_1 p_1 \tag{23}$$

with degradation rate k_{deg}, Hes1 mRNA concentration m, cytosolic Hes1 protein concentration p_1, and nuclear Hes1 protein concentration p_2. The parameters $\boldsymbol{\theta} = \{p_0, h, k_1, \nu\}$ govern the dynamics of the differential equations, and we estimate them in the unbounded $\log$ space to facilitate inference. p_0 corresponds to the amount of Hes1 protein in the nucleus when the rate of transcription of Hes1 mRNA is at half of its maximum value, h is the Hill coefficient, k_1 is the rate of transport of Hes1 protein into the nucleus and ν is the rate of translation of Hes1 mRNA (Silk et al., 2011).

In accordance to (Filippi et al., 2011), we use fixed initial conditions $m_0 = 2, p_1 = 5, p_2 = 3$ and set $k_{reg} = 0.03$. In our model we regard the observed mRNA concentrations $\mathbf{y}_t$ as noisy measurements of the true underlying mRNA concentration m_t with unit Gaussian observation error, $\mathbf{y}_t \sim \mathcal{N}(m_t, 1)$.

Silk et al. (Silk et al., 2011) used quantitative real-time PCR to collect the real experimental data

$$\mathbf{Y} = [1.20, 5.90, 4.58, 2.64, 5.38, 6.42, 5.60, 4.48]$$

where the first observation $\mathbf{y}_1$ is measured after 30 minutes, and all subsequent values are measured in 30 minute intervals (Filippi et al., 2011). The mRNA measures $\mathbf{y}_t$ refer to fold changes relative to a control sample. The Bayesian model uses Gamma priors on all parameters,

$$p_0 \sim \Gamma(2, 1), \quad h \sim \Gamma(10, 1), \quad k_1 \sim \Gamma(2, 50), \quad \nu \sim \Gamma(2, 50), \tag{24}$$

where $\Gamma(a, b)$ denotes the Gamma distribution with shape a and rate b.

Both NPLE (baseline) and our self-consistent approximator are trained for 70 epochs and use the same neural spline flow architecture consisting of 4 coupling layers with spectral normalization (Miyato et al., 2018) and a heavy-tailed Student-t_{50} latent distribution (Alexanderson & Henter, 2020) for a more robust latent space. For training, we use a batch size of 16 and an initial learning rate of $10^{-}3$. For the self-consistent approximator, we use $K = 100$ Monte-Carlo samples.

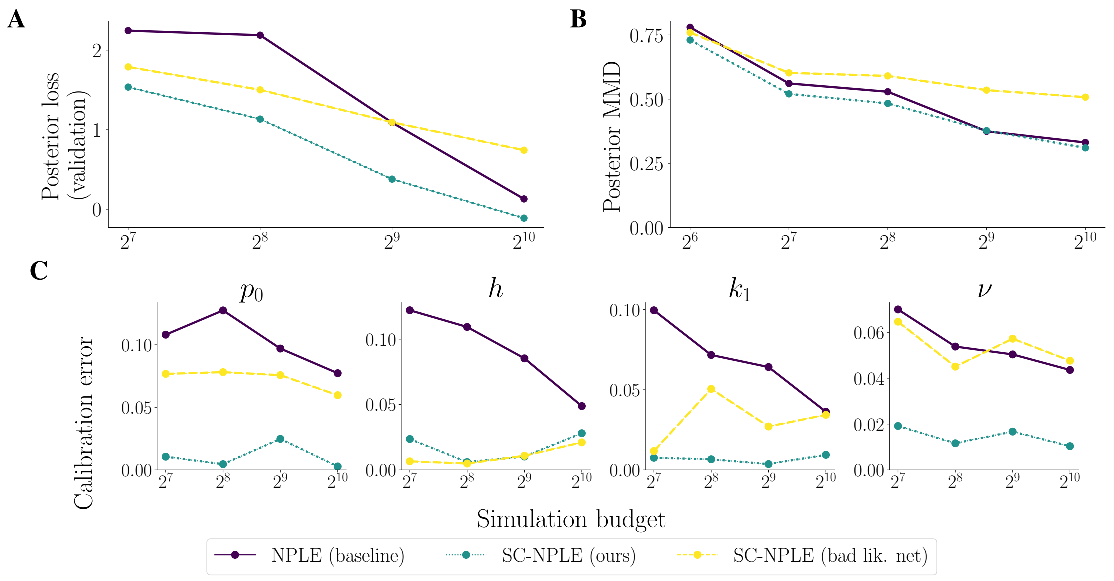


Figure 9: **Experiment 3, Ablation: Underexpressive likelihood network.** For the underexpressive likelihood network, we use a neural spline flow architecture with a single coupling layer, no L2 regularization, no dropout and a linear activation function. For low simulation budgets, posterior loss on a separate validation dataset is lower than standard NPLE (**A**). However, MMD between the approximate and true posterior is larger for SC-NPLE using the underexpressive likelihood network (**B**). The calibration error (**C**) lies between the calibration errors obtained from NPLE and SC-NPLE with a more suited likelihood network.

F. Details about Experiment 4

The inference task in this experiment is to locate a hidden source $\boldsymbol{\theta} \in \mathbb{R}^2$ from N noisy measurements $\mathbf{Y} \in \mathbb{R}^{N\times 1}$ of its signal intensity, which is observed at N pre-determined measurement points $\mathbf{X} \in \mathbb{R}^{\mathbb{N}}$ (see Figure 11).

More concretely, the sources is sampled from a standard Gaussian: $\boldsymbol{\theta} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(\boldsymbol{\theta} \,|\, \mathbf{0}, \mathbf{I})$ and the likelihood of the outcome is $\mathbf{Y} \sim \mathcal{N}(\mathbf{Y} \,|\, \nu(\boldsymbol{\theta}, \mathbf{x}), \sigma^2)$, where $\nu(\boldsymbol{\theta}, \mathbf{X}) = b + \frac{\alpha}{(m+||\boldsymbol{\theta}-\mathbf{X}||)^2}$. For convenience, we concatenate the measurements $\mathbf{Y}$ and measurement point locations $\mathbf{X}$ to a matrix of observables with $(N \times 3)$ elements. In the given context, α may be either predetermined constants or random variables, $b > 0$ represents a fixed background signal, and m is a constant representing the maximum signal. In our experiment we use $\sigma = 0.5, \alpha = 1, b = 0.1$ and $m = 10^{-4}$.

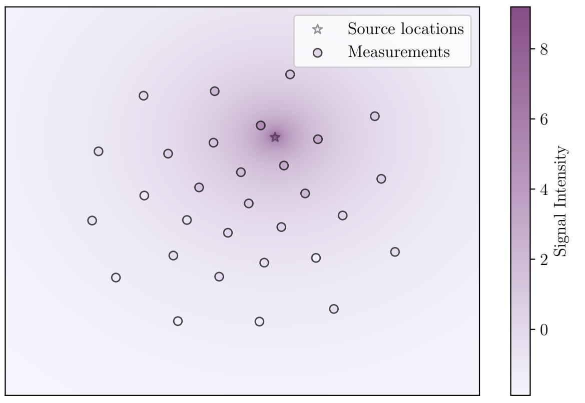


Figure 11: **Experiment 4 (Source Location Finding).** Illustration of the problem setup.

Both baseline NPE and our self-consistent approximator (SC-NPE) use identical neural networks and hyperparameters to ensure a fair comparison. We use an attention-based permutation-invariant neural network, i.e., a set transformer (Lee et al., 2019), to learn 32-dimensional embeddings that are maximally informative for posterior inference (Radev et al., 2020). The posterior network q_ϕ is a neural spline flow (Durkan et al., 2019) with 6 coupling layers. The neural networks are trained for 35 epochs with an initial learning rate of 10^{-3} and a batch size of 32.

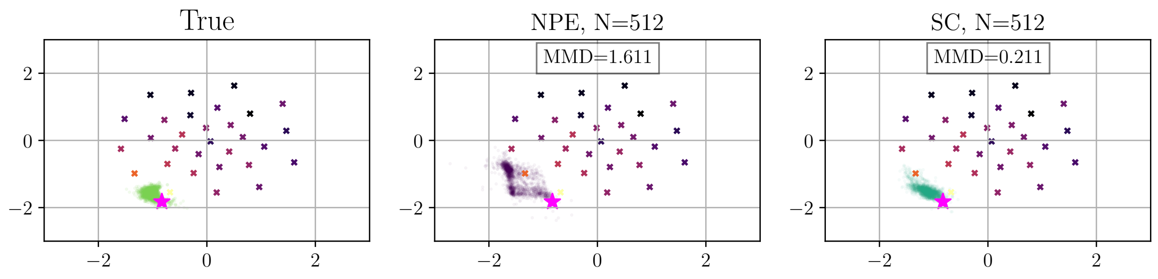


Figure 10: **Experiment 4 (Source Location Finding).** For one fixed data set, we show the reference posterior (HMC via Stan), as well as NPE (baseline) and SC-NPE (ours). Compared to the NPE baseline, our self-consistent approximator yields a noticeably sharper posterior without introducing additional bias, which is supported by a lower (better) MMD to the reference posterior.

G. Details about Experiment 5

The experiment setup follows Lueckmann et al. (2021) (Section T.9), with a modification: we use 160 instead of 10 evenly spaced time points. The neural network architectures are identical for NPLE (baseline) and SC-NPLE (ours). The posterior and likelihood networks also share same architectures: we use neural spline flow (Durkan et al., 2019) with 5 coupling layers. The latent space of the flow is a Student-t distribution with 30 degrees of freedom. We train for 100 epochs, batch size of 32 and initial learning rate of 10^{-3}, simultaneously learning the amortized likelihood and posterior. For the self-consistent variations, we apply a piecewise constant schedule to the weight λ: $\lambda = 0$ for the first 2 epochs (i.e., no self-consistency term) and $\lambda = 0.1$ for the remaining epochs.