Master's thesis
Master's Programme in Data Science

Neural Amortization of Bayesian Point Estimation

Yinong Li

April 10, 2024

Supervisor(s): Assistant Professor Luigi Acerbi

Examiner(s): Assistant Professor Luigi Acerbi
Dr. Grégoire Clarté

University of Helsinki
Faculty of Science

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Degree programme
Faculty of Science	Master's Programme in Data Science

Tekijä — Författare — Author

Yinong Li

Työn nimi — Arbetets titel — Title

Neural Amortization of Bayesian Point Estimation

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidantal — Number of pages
Master's thesis	April 10, 2024	56

Tiivistelmä — Referat — Abstract

The thesis is about developing a new neural network-based simulation-based inference (SBI) method for performing flexible point estimation; we call this method Neural Amortization of Bayesian Point Estimation (NBPE). Firstly, using neural networks, we can achieve amortized inference so that most of the computation cost is spent on training the neural network while performing inference only costs a few milliseconds. In this thesis, we utilize an encoder-decoder architecture; we use an encoder as a summary network to extract informative features from raw data and then feed them to a decoder as an inference network to output point estimations. Moreover, with a novel training method, the utilization of a variable α in the loss function $|\theta_i - \theta_{\text{pred}}|^\alpha$ enables the prediction of different statistics (mean, median, mode) of the posterior distribution. Thus, with our method, at inference time, we can get a fast point estimation, and if we want to get different statistics of the posterior, we have to specify the value of the power of the loss α. When $\alpha = 2$, the result will be the mean; when $\alpha = 1$, the result will be the median; and when α is getting closer to 0, the result will approach the mode.

We conducted comprehensive experiments on both toy and simulator models to demonstrate these features. In the first part of the analysis, we focused on testing the accuracy and efficiency of our method, NBPE. We compared it to the established method called Neural Posterior Estimation (NPE) in the BayesFlow SBI software. NBPE performs with competitive accuracy compared to NPE and can perform faster inference than NPE. In the second part of the analysis, we concentrated on the flexible point estimation capabilities of NBPE. We conducted experiments on three conjugate models since most of these models' posterior mean, median, and mode have analytical expressions, which leads to more straightforward analysis. The results show that at inference time, the different choices of α can influence the output exactly, and the results align with our expectations.

In summary, in this thesis, we propose a new neural SBI method, NBPE, that can perform fast, accurate, and flexible point estimation, broadening the application of SBI in downstream tasks of Bayesian inference.

ACM Computing Classification System (CCS):
Mathematics of computing → Probability and statistics → Probabilistic inference problems→ Bayesian computation
Computing methodologies → Machine learning → Machine learning approaches → Neural networks
Computing methodologies → Machine learning → Learning paradigms → Supervised learning → Supervised learning by regression

Avainsanat — Nyckelord — Keywords

Bayesian inference, point estimation, amortized inference, neural network

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Preface

First and foremost, I would like to express my deepest gratitude to my supervisors, Prof. Luigi Acerbi and Dr. Grégoire Clarté, for their support, feedback, and suggestions throughout this work. I am also indebted to M.Sc. Daolang Huang, whose profound insights into my work and guidance in structuring this thesis have been invaluable. Finally, I would like to acknowledge the usage of Grammarly and ChatGPT4 to enhance the quality of this thesis, including grammar and spell-checking.

Helsinki, April 6, 2024,

Yinong Li

Contents

1. Introduction

1.1 Motivation

Stochastic numerical simulators have become instrumental in many fields, from the sciences to economics, providing insights into understanding complex systems [Cranmer et al., 2020, Ratmann et al., 2007, Geyer, 1992]. A crucial and challenging task in utilizing these simulators is inferring the model parameters given observed data in these models (also called simulator models). In the context of Bayesian analysis, this translates to inferring the posterior distribution. Conventional Bayesian inference methods, such as Variational Inference (VI) and Markov Chain Monte Carlo (MCMC), are likelihood-based. However, they are not very useful for this kind of model since the simulator models' likelihood functions are either intractable or evaluating the likelihood is computationally extremely expensive, leading to infeasible inference [Lückmann, 2022, Cranmer et al., 2020].

A new class of methods, likelihood-free or simulation-based inference (SBI), has been developed in this context. SBI offers an alternative framework for Bayesian inference, especially in the context of simulator models with intractable likelihoods. It bypasses the need for explicit likelihood calculations. Instead, it uses simulated data to approximate the likelihood and infer the posterior with the prior or directly approximate the posterior [Cranmer et al., 2020, Sisson et al., 2018]. Much research has shown that SBI is particularly advantageous for non-differentiable models or containing inaccessible internal random variables. This has facilitated exploration within complex simulation domains, supported by the development of efficient algorithms and toolboxes that streamline statistical inference after simulation-based training [Radev et al., 2020, Tejero-Cantero et al., 2020]. Its applications have proliferated, contributing significantly to fields such as neuroscience, particle physics, epidemiology, and cosmology [Cranmer et al., 2020].

Within the field of SBI, initial methods like Approximate Bayesian Computation (ABC) have paved the way for understanding complex stochastic systems through likelihood-free inference [Beaumont et al., 2002]. For example, the most classical ABC method, Rejection-ABC, employs a rejection-sampling method to avoid evaluating the

likelihood. However, these approaches often involve computationally demanding procedures for each new dataset and struggle with high-dimensional problems due to the high computational cost of the rejection sampling process. With the development of machine learning, especially deep learning, the field has since progressed with neural network-based amortized inference techniques, such as Neural Posterior Estimation (NPE), Neural Likelihood Estimation (NLE), and Neural Ratio Estimation (NRE). Unlike REJ-ABC, these neural network-based approaches can scale better to high-dimensional problems and amortize the inference process, allowing for efficient generalization across different instances and reducing the computational burden for new observations [Lueckmann et al., 2021, Cranmer et al., 2020].

These SBI approaches, however, typically aim to approximate the entire posterior or likelihood, which may not be the most efficient strategy for all applications. The real world often demands faster, targeted inferences, especially in systems modeled by complex simulators, where rapid point estimation is vital. In these cases, direct output of the desired statistics, bypassing the intermediate step of complete posterior approximation, can significantly expedite the process. Furthermore, the training of neural networks for point estimation can be treated as a special case Bayesian decision-making framework. This perspective shifts the focus to identifying the most informative point estimates based on the simulations, which are similar to finding optimal actions in a decision-making scenario. This approach is significant given that some neural network-based SBI methods can produce only approximate posteriors that might lead to suboptimal decisions [Gorecki et al., 2023, Alsing et al., 2023, Lacoste-Julien et al., 2011]. By adopting a direct point estimation approach, we aim to achieve more accurate decision-making, reflecting a more precise understanding of the underlying models.

1.2 Contributions

This thesis proposes 'Neural Amortization of Bayesian Point Estimates (NBPE)', an innovative neural-network SBI method that adapts amortized inference to do a special downstream task of Bayesian inference, point estimation. Similar to other Neural SBI methods, NBPE can also amortize the cost of inference by leveraging the features of neural networks; however, unlike other Neural SBI methods, it should always first approximate posterior and then get point estimations, which could be inefficient. NBPE trains an end-to-end network to output accurate point estimation directly in milliseconds.

A novel feature of our approach is the implementation of a variable loss function $|\theta_i - \theta_{\text{pred}}|^\alpha$, where the value of α is varied during each training epoch, the θ_i is the parameter proposed by $p(\theta)$ and the θ_{pred} is our predicted result. With this training

method, at inference time, by varying the value of α, we can get the statistics of the posterior we want. Typically, when $\alpha = 2$, we are supposed to get the mean of the posterior. When $\alpha = 1$, we are supposed to get the median of the posterior. When α is close to 0, we are approaching the mode of the posterior. This feature makes NBPE output accurate and flexible regarding point estimation at inference time.

The efficacy of this method is demonstrated through comprehensive experimental analysis, which is divided into two parts. The first part focuses on evaluating the accuracy and efficiency of NBPE, testing it on both a toy model and a simulator model. We compared it to the established method called Neural Posterior Estimation (NPE) in the BayesFlow SBI software. NBPE performs with competitive accuracy compared to NPE and can perform faster inference than NPE. The second part of the experimental analysis concentrates on evaluating the flexible point estimation capabilities of NBPE on three different models: Poisson-Gamma, Beta-Bernoulli, and Gamma-Exponential. This part of the analysis aims to showcase the model's ability to target specific point estimates effectively across various probability distributions.

Additionally, our work illuminates the potential of viewing point estimation as a decision-making process within the Bayesian Decision Making (BDM) framework. By framing point estimation in this manner, NBPE simplifies the inference process and enriches the toolkit for downstream BDM tasks, offering novel insights and methodologies for SBI methods used in BDM.

In summary, the contributions of this research are manifold:

1. The application of amortized inference for efficient point estimation in SBI.

2. The implementation of a variable loss function training strategy to enable rapid and specific posterior statistic estimation.

3. Comprehensive experiments showing that the model's effectiveness for both toy models and simulator models.

4. Providing a new perspective on point estimation as a BDM process, potentially enhancing decision-making in downstream applications.

1.3 Outline

The thesis is structured as follows: Chapter 2 provides the necessary background to understand our work, beginning with an overview of Bayesian inference fundamentals, including basic ideas of Bayesian inference, likelihood-based inference methods, and a classical likelihood-free (simulation-based) inference method, Approximate Bayesian

Computation (ABC). The discussion then covers neural networks, the basis for amortized simulation-based inference, specifically introducing feedforward networks and the DeepSets architecture used in this research. A comprehensive overview of Simulation-Based Inference (SBI) is then presented, starting with Neural Posterior Estimation, followed by Neural Likelihood Estimation, and then Neural Ratio Estimation. Section 3 reviews related literature, focusing on existing works regarding neural point estimation and simulation-based inference methods in the context of Bayesian Decision Making. Section 4 details the methodology behind our proposed Neural Bayesian Point Estimation (NBPE) approach. This includes explaining the encoder-decoder structure and the novel variable loss function, as well as the training process and how it aligns with the Bayesian Decision Making framework. Section 5 describes the experiments conducted to validate the NBPE method. Section 6 discusses the limitations of the current research and potential future work, and summarizes the thesis. The Appendix provides additional information to support the main content of the thesis.

2. Preliminaries

In this chapter, we cover the fundamental concepts necessary to understand our work. We begin by introducing Bayesian Inference, a framework that allows us to update our beliefs about model parameters based on observed data. This introduction includes the Bayes Theorem, likelihood-based methods, such as with conjugate priors or variational inference, and finally, we cover a traditional likelihood-free or simulation-based inference method, Approximate Bayesian Computation (ABC). Next, we cover neural networks, which play an essential role in advanced simulation-based inference methods and our work. Finally, we discuss how neural networks can be utilized to develop more advanced Simulation-Based Inference (SBI) techniques, such as Neural Posterior Estimation (NPE), Neural Likelihood Estimation (NLE), and Neural Ratio Estimation (NRE).

2.1 Bayesian inference

In this section, we introduce the fundamental concepts of Bayesian inference, a framework for updating beliefs about model parameters based on observed data. We present Bayes' rule, which forms the foundation for computing the posterior distribution given the prior and the likelihood. We also discuss conjugate priors, Variational Inference (VI), and Approximate Bayesian Computation (ABC) as alternative inference techniques.

2.1.1 Definition

In the field of statistics, there are two distinct classes of methods: Frequentist and Bayesian. Frequentists believe that parameters exist objectively and are fixed but unknown values. In contrast, Bayesians do not focus on the correct parameter values but instead aim to derive posterior probabilities for statistical inference by incorporating observed data with prior knowledge [Betancourt, 2019, Gelman et al., 1995].

In a nutshell, the Bayesian method uses prior knowledge of the unknown parameters θ, combined with the currently observed data $\mathbf{x}_o$, to update the belief about

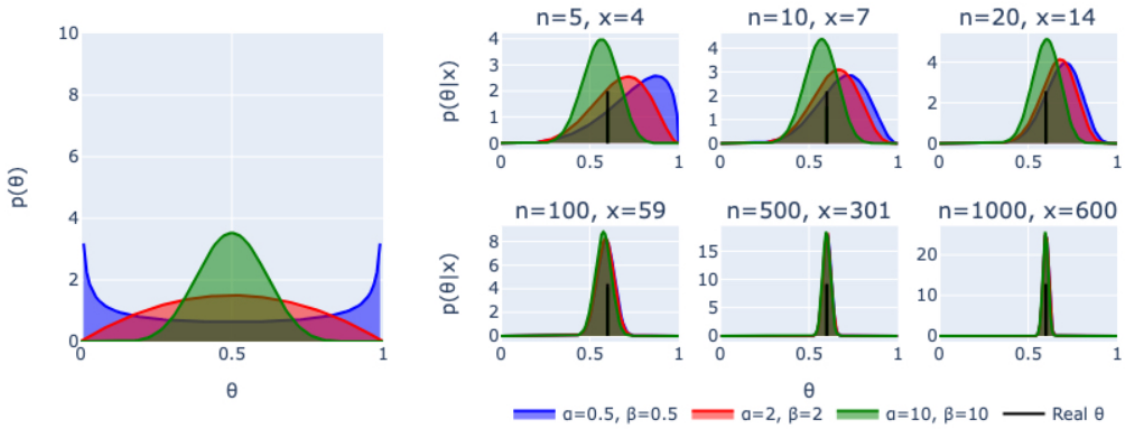


Figure 2.1: This set of plots demonstrates the evolution of the posterior distribution of the probability parameter θ (representing the likelihood of a coin landing heads up) as observed data is accumulated. Each subplot corresponds to a different number of coin toss trials (n = 5, 10, 20, 100, 500, 1000) with the respective number of heads observed ($\mathbf{x}$). The posterior distributions $p(\theta|\mathbf{x})$ are computed using three different sets of Beta distribution priors: $\alpha = 0.5, \beta = 0.5$ (blue), $\alpha = 2, \beta = 2$ (green), and $\alpha = 10, \beta = 10$ (red). The true value of θ (0.6) is marked by a vertical black line in each plot. These visualizations show how the choice of prior influences the posterior distribution, especially with a smaller number of observations, and how the posterior distribution converges towards the true value as more data is observed.

unknown parameters, which is the posterior probability. This updating process is clearly illustrated by Bayes' Theorem:

$$\underbrace{p(\theta|\mathbf{x}_o)}_{\text{posterior}} = \frac{p(\theta, \mathbf{x}_o)}{p(\mathbf{x}_o)} = \frac{\overbrace{p(\mathbf{x}_o|\theta)}^{\text{likelihood}}\overbrace{p(\theta)}^{\text{prior}}}{\underbrace{p(\mathbf{x}_o)}_{\text{evidence}}}, \text{ where } p(\mathbf{x}_o) = \int p\left(\mathbf{x}_o|\theta'\right) p\left(\theta'\right) \mathrm{d}\theta'. \tag{2.1}$$

This formula encapsulates the essence of Bayesian learning, where the prior probability $p(\theta)$ is updated to the posterior probability $p(\theta|\mathbf{x})$ after considering the observed data. Such an approach not only allows for the incorporation of prior knowledge but also facilitates a more comprehensive assessment of uncertainty, distinguishing Bayesian methods from traditional frequentist approaches in statistical modeling [Tipping, 2004]. We illustrate the Bayesian updating process in Figure 2.1.

2.1.2 Conjugate priors

After establishing the core principle of Bayesian inference (deriving the posterior distribution using Bayes' theorem), the next challenge is implementing this process efficiently. The computational complexity of updating beliefs about unknown parameters becomes particularly evident when the model and data are complex. In such cases, conjugate priors are critical in simplifying the Bayesian updating process. A prior is said

to be conjugate to the likelihood function if the posterior distribution is in the same family as the prior distribution. This relationship allows for an analytical solution to the Bayesian updating process, making the inference computationally straightforward.

For example, consider a coin toss experiment. The probability of the coin landing heads up is denoted by θ. Our goal is to infer the value of θ based on observed coin toss outcomes. We express our prior belief about θ using a Beta distribution:

$$p(\theta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}, \tag{2.2}$$

where α and β are the shape parameters of the Beta distribution.

The likelihood of observing $\mathbf{x}_o$ heads in n tosses is given by the binomial distribution:

$$p(\mathbf{x_o}|\theta) = \frac{n!}{\mathbf{x_o}!(n-\mathbf{x_o})!}\theta^{\mathbf{x_o}}(1-\theta)^{n-\mathbf{x_o}}. \tag{2.3}$$

By Bayes' Theorem, the posterior distribution is proportional to the product of the prior and the likelihood:

$$p(\theta|\mathbf{x}_o) \propto \theta^{\alpha-1}(1-\theta)^{\beta-1}\theta^{\mathbf{x}}(1-\theta)^{n-\mathbf{x}_o} = \theta^{\alpha+\mathbf{x}_o-1}(1-\theta)^{\beta+n-\mathbf{x}_o-1}. \tag{2.4}$$

Thus, the posterior parameters become:

$$\alpha_{\text{posterior}} = \alpha + \mathbf{x}_o, \quad \beta_{\text{posterior}} = \beta + n - \mathbf{x}_o. \tag{2.5}$$

This shows that by using a Beta prior, the posterior also follows a Beta distribution. The conjugacy ensures that Bayesian inference can be performed efficiently, without the need for complex integrations or numerical approximations.

Challenges While conjugate priors simplify the computation of the posterior, real-world applications often do not present such convenient scenarios. Recall the equation:

$$p(\theta|\mathbf{x}_o) \propto p(\mathbf{x}_o|\theta)p(\theta), \tag{2.6}$$

in the absence of conjugate priors, the posterior $p(\theta|\mathbf{x}_o)$ does not conform to a recognizable distribution form, complicating its direct computation. Specifically, the denominator of Bayes' theorem, the marginal likelihood:

$$p(\mathbf{x}_o) = \int p(\mathbf{x}_o|\theta)p(\theta)d\theta, \tag{2.7}$$

becomes intractable due to the integral over θ, which may involve exponential complexity in high-dimensional spaces or when the model structure does not allow for simplification[Gelman et al., 1995]. Consequently, the absence of an analytical solution for the posterior necessitates alternative computational strategies to approximate it effectively.

Solutions This computational complexity sets the stage for the adoption of advanced methods such as Markov Chain Monte Carlo (MCMC) and Variational Inference (VI). MCMC circumvents the direct calculation by constructing a Markov Chain to sample from the posterior distribution, providing a way to approximate it in complex scenarios [Murray, 2007]. While MCMC is known for its accuracy, it can be computationally demanding [Wiqvist et al., 2021].

Conversely, Variational Inference offers a more computationally efficient alternative by approximating the posterior with a tractable surrogate distribution. This approximation turns the problem of Bayesian inference into an optimization problem, which is particularly advantageous in handling large-scale data and complex models [Zhang et al., 2018, Blei et al., 2017]. In the next section, we will cover Variational Inference, discussing its methodology, classic algorithm, and challenges when the likelihood function is intractable.

2.1.3 Variational inference

Idea Variational Inference (VI) is a computational technique in Bayesian inference that approximates complex posterior distributions $p(\theta|\mathbf{x}, \phi)$ with a simpler, parameterized distribution $q(\theta|\mathbf{x}, \lambda)$. The core idea involves selecting a tractable approximate distribution $q(\theta|\mathbf{x}, \lambda)$ and optimizing its parameters λ to minimize the discrepancy between $q(\theta|\mathbf{x}, \lambda)$ and the true posterior $p(\theta|\mathbf{x}, \phi)$.

Problem Reformulation When using the Kullback-Leibler (KL) divergence as the distance measure, the optimization objective becomes:

$$\lambda^* = \arg\max_{\lambda} \underbrace{E_{\theta\sim q(\theta|\mathbf{x},\lambda)}[\log p(\mathbf{x}|\theta,\phi)] - E_{\theta\sim q(\theta|\mathbf{x},\lambda)}\left[\log \frac{q(\theta|\mathbf{x},\lambda)}{p(\theta)}\right]}_{ELBO(q)} \tag{2.8}$$

This formulation, known as the Evidence Lower Bound (ELBO), transforms the intractable problem into a computable optimization objective involving the tractable form of $q(\theta|\mathbf{x}, \lambda)$, the known likelihood function $p(\mathbf{x}|\theta, \phi)$, and a predefined prior $p(\theta)$ from the Bayesian model. For a detailed derivation of the ELBO, refer to Appendix A.2.

Coordinate Ascent Variational Inference (CAVI) CAVI is a classic VI approach that uses a mean-field variational family to describe $q(\theta|\mathbf{x}, \lambda)$ and coordinate ascent to solve the ELBO maximization problem. The mean-field variational family assumes independence among the hidden variables, leading to a factorized form of $q(\theta|\mathbf{x}, \lambda)$. CAVI then optimizes each factor $q_j(\theta_j|\mathbf{x}, \lambda_j)$ iteratively until the ELBO converges, effectively finding the optimal variational distribution $q(\theta|\mathbf{x}, \lambda)$. The efficacy of this

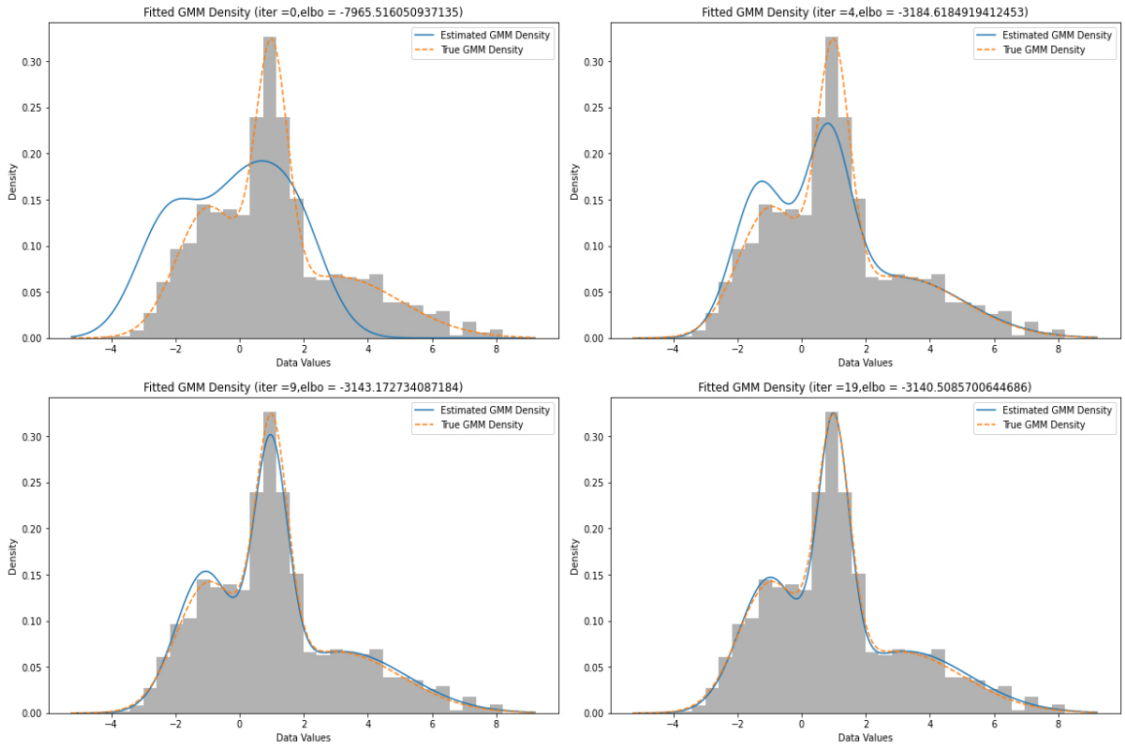


Figure 2.2: This figure shows the monotonic increase in ELBO that is indicative of the variational distribution's convergence towards the true posterior.

approach is illustrated through an application to a three-component Gaussian Mixture Model, as detailed in Figure 2.2, showing the iterative refinement of the variational parameters and the corresponding improvement in ELBO. A set of papers research more advanced VI methods from different angles. For example, applying more complex prior distributions has facilitated a richer characterization of parameter information. See these papers if you are interested [Tomczak and Welling, 2018, Atanov et al., 2019].

2.1.4 Challenges with Likelihood-Based Inference Methods

From the above introduction, methods like VI offer a promising strategy for Bayesian inference when dealing with complex models by approximating the intractable true posterior with a tractable variational distribution. Such approaches are also called likelihood-based inference methods generally due to their reliance on the direct evaluation of the likelihood $p(\mathbf{x_o}|\theta)$. Although they could be very efficient on some tasks of Bayesian inference, the reliance on direct likelihood evaluation makes MCMC and VI less feasible in scenarios where the likelihood function is intractable [Lintusaari et al., 2017]. This challenge is prevalent in many scientific and engineering fields where mod-

els have intractable likelihood functions, limiting the applicability of VI and MCMC in such contexts [Karabatsos and Leisen, 2018]. An example of such cases is Simulator Models.

Simulator Models Simulator models, also known as implicit models, are a class of computational frameworks that define the data generation process without an explicitly defined likelihood function. These models typically appear as computer programs and are important and widespread in various scientific fields such as particle physics [Butanovs et al., 2021], astronomy [Alsing et al., 2018], neuroscience [Lueckmann et al., 2017], and population genetics [Beaumont et al., 2002], where they are employed to construct sophisticated, high-fidelity simulations with the support of modern computational resources.

Although indispensable in many scientific fields, simulator models present challenging issues in Bayesian statistical analysis due to their inexplicit likelihood functions. This characteristic often leads to the intractability of likelihood computation, either because the evaluations are too expensive or because explicit evaluations are simply impossible [Cranmer et al., 2020, Papamakarios, 2019]. Such intractability significantly hinders the application of likelihood-based Bayesian inference methods such as MCMC and VI [Papamakarios and Murray, 2016].

These limitations have motivated the development of simulation-based inference (SBI) methods, which circumvent the need for evaluating the likelihood explicitly. In the following subsections, we will introduce some typical methods in simulation-based inference, also known as likelihood-free inference, to see how these methods can perform inference under intractable likelihood.

2.1.5 Approximate Bayesian Computation

To address the challenges posed by simulator models, alternative likelihood-free inference methods have been developed. One such approach is Approximate Bayesian Computation (ABC).

Idea The intractability of directly computing the likelihood $p(\mathbf{x} = \mathbf{x}_o|\theta)$ for simulator models motivated the development of ABC, a classical likelihood-free inference method. ABC circumvents the direct evaluation of the likelihood function by instead comparing observed data, $\mathbf{x}_o$, with simulated data, $\mathbf{x}_s$, using a norm function $d(\mathbf{x}_s, \mathbf{x}_o)$ to measure their distance. Parameter samples, θ', that result in simulated data closely matching the observed data are considered as samples from an approximate posterior distribution. Through iterative sampling, ABC accumulates a collection of such

samples, denoted by $\theta' \sim \hat{p}(\theta|\mathbf{x})$, where $\hat{p}$ represents the ABC-approximated posterior distribution [Sisson et al., 2018, Lintusaari et al., 2017].

Summary statistics From the earliest work on ABC, the use of summary statistics-informative features extracted from the data has been a cornerstone of the ABC methodology [Beaumont et al., 2002]. To manage the challenges posed by high-dimensional data, ABC typically does not directly utilize raw data, $\mathbf{x}$, but instead computes and utilizes summary statistics, $S(\mathbf{x})$, for comparison of the pseudo-observations. This approach addresses the 'curse of dimensionality': in high-dimensional settings, particularly those encountered in real-world models, the sample space expands exponentially with each additional dimension. This exponential growth makes the acceptance rate of ABC very low since it requires an impractically large number of simulations to thoroughly explore the parameter space and find parameter values that generate simulated data closely matching the observed data[Pacchiardi and Dutta, 2022].

The use of summary statistics, $S(\mathbf{x})$, helps reduce the data's dimensionality, therefore mitigating the curse of dimensionality. Summary statistics are carefully chosen low-dimensional representations that aim to capture the most relevant information from the high-dimensional data while discarding redundant or less informative details. By comparing these lower-dimensional summary statistics instead of the full data, the distance computation becomes more tractable and requires fewer simulations.

It is noteworthy that the concept of summary statistics is not unique to ABC but is a fundamental element across almost all simulation-based inference methods. In this thesis, for clarity and simplicity in subsequent discussions, we will use $\mathbf{x}$ to refer to $S(\mathbf{x})$ unless specified otherwise.

Rejection ABC For more detail, We will cover a classical ABC methodology, Rejection ABC, which implements the foundational principles of ABC through an acceptance-rejection scheme. This method operates on the premise of comparing observed data, $\mathbf{x}_o$, against simulated data generated from a prior distribution over the parameters, $p(\theta)$, of a simulator model $p(\mathbf{x} \mid \theta)$. Employing an l_2-norm function $d(\cdot, \cdot)$ and setting an acceptance threshold ϵ, Rejection ABC selectively accepts parameter samples θ' that yield simulated data closely resembling the observed data, as described in Algorithm 1.

Distinctively, Rejection ABC circumvents the direct computation of the likelihood function $p(\mathbf{x} = \mathbf{x}_o \mid \theta)$ by constructing a surrogate likelihood. This surrogate is derived from the integral across the entire data space $\mathbf{x}$, modulated by an indicator function I, which assigns a value of 1 when the simulated data $\mathbf{x}_s$ falls within an ϵ-radius of the

Algorithm 1 Classical REJ-ABC

Initial: A simulator $p(\mathbf{x}|\theta)$, a prior distribution $p(\theta)$, an observation $\mathbf{x_o}$, a tolerance threshold ϵ
while in simulation budget **do**
  Sample θ' from $p(\theta)$
  Simulate $\mathbf{x_s}$ from $p(\mathbf{x}|\theta')$
  if $\|\mathbf{x}_s - \mathbf{x}_o\| \leq \epsilon$ **then**
    Accept θ'
  else
    Reject θ'
  end if
end while
return Accepted samples $\{\theta'\}$ from $p\left(\theta | \|\mathbf{x}_s - \mathbf{x}_o\| \leq \epsilon\right)$

observed data $\mathbf{x}_o$, and 0 otherwise:

$$p_\epsilon\left(\mathbf{x}_o|\theta\right) = \int p(\mathbf{x}|\theta) I\left(\|\mathbf{x}_s - \mathbf{x}_o\| \leq \epsilon\right) d\mathbf{x}. \tag{2.9}$$

Here, $p(\mathbf{x}|\theta)$ is the likelihood of generating data $\mathbf{x}$ under parameter θ. Recall the Bayes' theorem Equation 5.2, we can integrate the surrogate likelihood $p_\epsilon\left(\mathbf{x}_o \mid \theta\right)$ with the prior $p(\theta)$ to formulate the ABC posterior $\hat{p}\left(\theta|\mathbf{x}_o\right)$:

$$\hat{p}\left(\theta|\mathbf{x}_o\right) \propto p_\epsilon\left(\mathbf{x}_o|\theta\right) \cdot p(\theta) \propto \left(\int p(\mathbf{x}|\theta) I\left(\|\mathbf{x}_s - \mathbf{x}_o\| \leq \epsilon\right) d\mathbf{x}\right) \cdot p(\theta). \tag{2.10}$$

In theory, given an unlimited simulation budget and as $\epsilon \to 0$, the approximate posterior $\hat{p}\left(\theta|\mathbf{x}_o\right)$ converges to the exact posterior $p\left(\theta \mid \mathbf{x} = \mathbf{x}_o\right)$, a convergence supported by detailed proofs in the literature [Prangle, 2017].

In sum, classical REJ-ABC is a basic implementation of the ABC framework that provides a straightforward approach to approximate the posterior distribution without requiring the evaluation of the likelihood function.

Related work The simplicity of Classical REJ-ABC, while straightforward, is not without limitations, particularly when the prior distribution significantly diverges from the posterior. This discrepancy often results in inefficient sampling, necessitating an excessive number of iterations to achieve convergence [Lückmann, 2022, Papamakarios, 2019]. Innovations in ABC methodologies have sought to mitigate this inefficiency by utilizing previously accepted parameters to iteratively refine the proposal distribution. Methods such as Markov Chain Monte Carlo ABC (MCMC-ABC) leverage Metropolis-Hastings algorithms to enhance parameter proposals, thereby improving acceptance rates [Marjoram et al., 2003]. Similarly, Sequential Monte Carlo ABC (SMC-ABC)

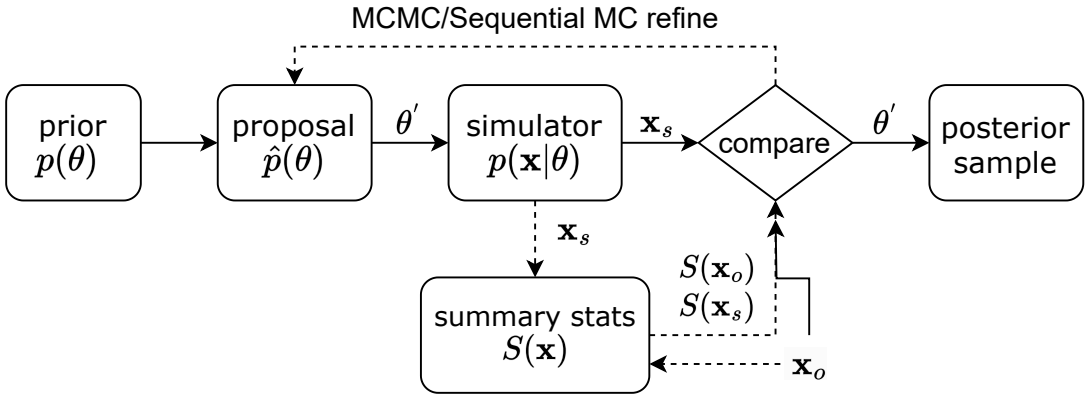


Figure 2.3: Overview of different REJ-ABC algorithms.

employs importance sampling to incrementally sample from progressively refined posterior distributions, effectively decreasing the tolerance threshold ϵ and enhancing the efficiency of the sampling process [Bonassi and West, 2015, Beaumont et al., 2009, Sisson et al., 2007]. These advancements represent a significant evolution from the classical approach, offering more sophisticated mechanisms for parameter sampling and posterior approximation in ABC frameworks.

Limitations Advanced ABC algorithms like Sequential Monte Carlo ABC (SMC-ABC) have significantly improved acceptance rates over classical ABC by utilizing iteratively refined proposal priors. Despite these advancements, a fundamental challenge persists: the acceptance rate diminishes as the tolerance threshold ϵ decreases, particularly in high-dimensional data scenarios. The consequence is a dramatic increase in the required number of simulations, potentially necessitating hundreds of thousands of simulator calls for a single posterior sample, rendering the approach impractical for many applications [Geffner et al., 2023, Greenberg et al., 2019].

The utilization of summary statistics $S(\mathbf{x})$ instead of the raw data $\mathbf{x}$ can lead to highly accurate posterior approximations, provided that these statistics capture the essential information. If, however, the chosen summary statistics lack sufficient informativeness, they can negatively impact the quality of the posterior approximation. The challenge of identifying effective summary statistics (strike a balance between low dimension and informativeness), which often relies on domain-specific knowledge, remains a significant area of research within the ABC literature [Charnock et al., 2018, Blum et al., 2013, Fearnhead and Prangle, 2012].

Furthermore, traditional ABC methods are inherently inefficient for datasets with numerous observations. The reliance on data in the rejection process, or in guiding the proposal distribution in more advanced algorithms, necessitates repeating the inference process for new observations. Consequently, ABC's suitability is confined mainly to

scenarios with single or few independent and identically distributed (i.i.d.) data points, limiting its applicability in broader contexts [Cranmer et al., 2020].

Recent years, due to the fast development of deep learning, advanced SBI methods have adopted neural networks to make SBI more scalable and efficient. In the next section, we cover some basic ideas of neural networks and introduce specific network architectures used in our work.

2.2 Neural networks

Neural networks also known as artificial neural networks (ANNs) have emerged as a powerful tool for approximating complex functions[LeCun et al., 2015], making them an indispensable component of many modern machine learning and inference methods, including SBI. At their core, neural networks are a class of flexible function approximators that can learn to map input data to desired outputs through a process of training on examples. In this section, we will focus on Feedforward Neural Networks, the most fundamental type of neural networks, to introduce the core concepts of neural network. Following this introduction, we will explore DeepSets, a specific network architecture that is integral to our methodology, highlighting its unique features and applications.

2.2.1 Feedforward neural networks

Feedforward Neural Networks (FNNs) represent the most basic and widely understood class of neural networks. A FNN consists of layers of nodes, or "neurons", each layer fully connected to the next without any cycles or feedback loops-hence the term "feedforward". The architecture typically comprises an input layer, one or more hidden layers, and an output layer. Each neuron in a layer receives input from the neurons of the preceding layer, processes this input through a weighted sum followed by a non-linear activation function, and passes the result forward. Mathematically, a FNN can be described through its layers and the transformations applied at each stage. Given an input vector $\mathbf{x} \in \mathbb{R}^d$, where d is the dimensionality of the input, an FNN processes this input through L layers to produce an output $\mathbf{y}$. Each layer l in the network applies a weighted linear transformation followed by a non-linear activation function $\sigma(\cdot)$ to its input:

$$\mathbf{h}^{(l)} = \sigma\left(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}\right), \tag{2.11}$$

where $\mathbf{h}^{(0)} = \mathbf{x}$ is the input layer, $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ denote the weight matrix and bias vector for layer l, respectively, and $\mathbf{h}^{(l)}$ represents the output of layer l. The final layer (often without a non-linear activation for regression tasks or with a softmax activation

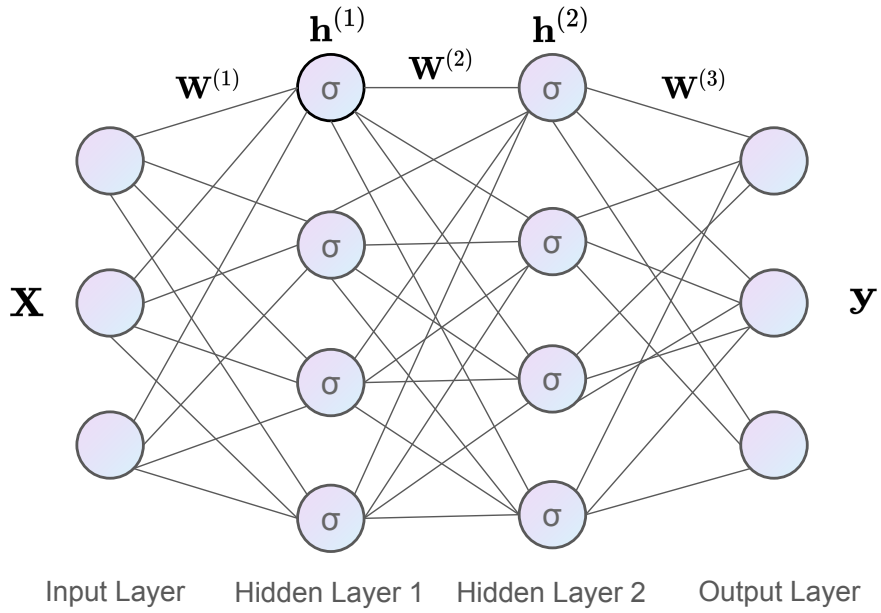


Figure 2.4: An architecture of a feedforward neural network with three layers, illustrating the connectivity from a 3-dimensional input to a 3-dimensional output through two 4-dimensional hidden layers. With the flow of information proceeding from left to right without any feedback loops, characteristic of feedforward neural networks.

for classification) produces the network's output:

$$\mathbf{y} = \mathbf{W}^{(L)}\mathbf{h}^{(L-1)} + \mathbf{b}^{(L)}. \tag{2.12}$$

Learning in FNNs involves adjusting $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ to minimize a loss function $\mathcal{L}(\mathbf{y}, \mathbf{y}_{\text{true}})$, where $\mathbf{y}_{\text{true}}$ is the true output. This is typically achieved using backpropagation to compute gradients and an optimization algorithm like stochastic gradient descent (SGD) [LeCun et al., 2015] for updates:

$$\mathbf{W}^{(l)} \leftarrow \mathbf{W}^{(l)} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(l)}}, \quad \mathbf{b}^{(l)} \leftarrow \mathbf{b}^{(l)} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(l)}}, \tag{2.13}$$

with η being the learning rate.

Activation functions $\sigma(\cdot)$, such as ReLU or sigmoid, introduce non-linearity, enabling FNNs to model complex relationships. Through this structured process, FNNs learn to approximate the function mapping inputs to outputs, showcasing their versatility across a range of tasks from classification to regression.

FNNs are straightforward and efficient, excelling in tasks like regression and classification with static inputs. However, their inability to process sequential data such as text and time series limits their application in areas where AI is rapidly advancing. To address these limitations, neural network architectures have evolved, introducing models like Recurrent Neural Networks (RNNs)[Rumelhart et al., 1986], Long Short-Term Memory networks (LSTMs)[Hochreiter and Schmidhuber, 1997], and Transformers for

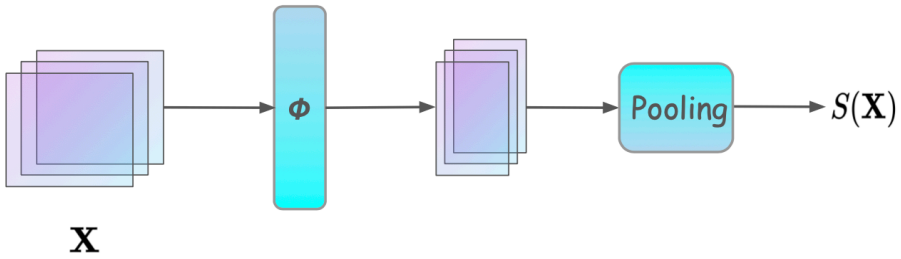


Figure 2.5: Simplified DeepSets architecture for permutation-invariant summary statistic extraction from simulation sets [Lee et al., 2018]. The set X undergoes a transformation via the ϕ function, followed by a pooling layer that computes sum or average to generate the set representation $S(X)$. This representation captures the essential information needed for further analysis, omitting the ρ function to focus on encoding through aggregation.

handling sequential tasks in Natural Language Processing (NLP), and Convolutional Neural Networks (CNNs)[Krizhevsky et al., 2012] for spatial tasks in computer vision. These advanced structures allow neural networks to interpret contextual and temporal information, expanding their utility across a wider range of applications.

2.2.2 DeepSets Framework

Permutation invariance is a crucial property in set data processing, referring to the condition where the order of inputs does not influence the output of a function. Formally, for a set of inputs $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ and a permutation π that rearranges the elements of X into $X_\pi = \left\{\mathbf{x}_{\pi(1)}, \mathbf{x}_{\pi(2)}, ..., \mathbf{x}_{\pi(n)}\right\}$, a function f is said to be permutation invariant if:

$$f(X) = f(X_\pi), \tag{2.14}$$

for all possible permutations π. This condition ensures that the result of applying f remains consistent regardless of the order of elements in X.

FNNs, such as Multilayer Perceptrons (MLPs), are designed to process inputs of a fixed size, producing outputs that correspond directly to the input data size. This one-to-one input-output mapping makes FNNs lack a mechanism to aggregate information across multiple input vectors while preserving the unordered nature of sets. And some advanced architectures such as CNNs and RNNs utilize on the order of inputs, rendering them sensitive to permutations. Consequently, they are not inherently suitable for tasks involving set data where order is not a defining characteristic. However, many real-world applications involve set-structured data, such as point cloud classification tasks in computer vision and robotics, and population statistics[Zaheer et al., 2017]. In these scenarios, the order of elements is irrelevant, and the primary focus is on capturing the overall properties of the set as a whole.

In response to this need, DeepSets provides a general framework for handling set

data in a permutation-invariant manner[Zaheer et al., 2017]. The key insight behind DeepSets is that any permutation-invariant function f over a set X can be decomposed into an element-wise transformation ϕ followed by a permutation-invariant aggregation operation (e.g., summation) and a final transformation ρ:

$$f(X) = \rho\left(\sum_{i=1}^{n} \phi(\mathbf{x}_i)\right). \tag{2.15}$$

The summation here serves as the permutation-invariant operation, satisfying the condition of permutation invariance:

$$\sum_{i=1}^{n} \phi\left(\mathbf{x}_i\right) = \sum_{i=1}^{n} \phi\left(\mathbf{x}_{\pi(i)}\right), \tag{2.16}$$

for any permutation π. This ensures that $f(X)$, the output of the DeepSets model, remains consistent regardless of the order of elements in X, embodying the principle of permutation invariance. It is worth noting that DeepSets proposes a generalized approach to processing ensemble data, where the transform function ϕ and the aggregation function ρ are not fixed in form. Considerable research has been done to represent these two classes of functions using stronger and more modern neural network architectures, such as the Attention-based SetTransformer[Lee et al., 2018]. Both DeepSets and SetTransformer play crucial roles in our proposed method, see Figure 2.5.

The expressive power and flexibility of neural networks make them particularly well-suited for learning complex posteriors in simulation-based inference problems. By incorporating neural networks into the inference process, we can potentially overcome the limitations of traditional methods and obtain more accurate and efficient approximations of the posterior distribution. In the following section, we will cover some advances neural network-based simulation inference methods.

2.3 Neural Simulation Based Inference

In recent years, by leveraging the powerful capabilities of neural networks, some advanced SBI methods have been developed to overcome the limitations mentioned in Section 2.1.5.

In this section, we will cover three typical neural network-based SBI methods in detail: Neural Posterior Estimation (NPE), Neural Likelihood Estimation (NLE), and Neural Ratio Estimation (NRE). We will introduce their methodologies, related works and limitations. For NPE and NLE, we will provide a more detailed discussion, while NRE will be briefly introduced.

2.3.1 Neural Posterior Estimation

The Neural Posterior Estimation (NPE) is initially proposed by Papamakarios and Murray [2016]. Unlike ABC, which relies on an accept-reject framework to approximate the posterior, NPE reformulates this problem to a density estimation problem: It leverages a neural network to serve as a conditional density estimator trained on simulated datasets to approximate the posterior.

Methodology The procedure begins with constructing a dataset $\mathcal{D} = \{(\theta_i, \mathbf{x}_i)\}_{i=1}^N$ through parameter sampling from the prior $p(\theta)$ and subsequent data generation via the simulator $p(\mathbf{x}|\theta)$. This dataset facilitates the training of a neural network-based density estimator $q_\psi(\theta|\mathbf{x})$, optimizing the neural network parameters ψ to minimizing loss function given by the the negative log probability $\log q_\psi(\theta|\mathbf{x})$, that is

$$\begin{aligned} \mathcal{L}(\psi) &= \mathbb{E}_{p(\theta)p(\mathbf{x}|\theta)} - \log q_\psi(\theta|\mathbf{x}) \\ &\approx \frac{1}{N} \sum_{i=1}^{N} \log q_\psi \left(\theta_i|\mathbf{x}_i\right). \end{aligned} \tag{2.17}$$

The idea here is to train the network to maximize the likelihood of the observed data under the density estimator $q_\psi(\theta|\mathbf{x})$, which corresponds to minimizing the forward Kullback-Leibler divergence $\mathrm{KL}(p(\theta|\mathbf{x})||q_\psi(\theta|\mathbf{x}))$, that is:

$$\begin{aligned} \psi^* &= \arg\min_\psi \mathbb{E}_{p(\mathbf{x})} \left[\mathrm{KL}\left(p(\theta|\mathbf{x}) \| q_\psi(\theta|\mathbf{x})\right)\right] \\ &= \arg\min_\psi \mathbb{E}_{p(\mathbf{x})} \mathbb{E}_{p(\theta|\mathbf{x})} \left[\log \frac{p(\theta|\mathbf{x})}{q_\psi(\theta|\mathbf{x})}\right] \\ &= \arg\min_\psi \mathbb{E}_{p(\theta,\mathbf{x})} \left[-\log q_\psi(\theta|\mathbf{x})\right], \end{aligned} \tag{2.18}$$

the amortization over $p(\mathbf{x})$ makes it possible to bypass the sampling or the evaluation of the unknown posterior $p(\theta|\mathbf{x})$ in the second line above. Indeed, the double expectation $\mathbb{E}_{p(\mathbf{x})}\mathbb{E}_{p(\theta|\mathbf{x})}$ can be rewritten as an expectation $\mathbb{E}_{p(\theta,\mathbf{x})}$ over the joint distribution, which we can easily sample in the forward direction as $p(\theta, \mathbf{x}) = p(\theta)p(\mathbf{x}|\theta)$ regardless of whether the likelihood is tractable or not [Vasist et al., 2023].

Then, at the inference phase, NPE efficiently computes the posterior estimate given any observations since now inference is just a simple forward pass, which significantly amortizes the computational expense of traditional SBI methods.

Related Work From the beginning of the first NPE algorithm to the present, researchers have developed many different versions of NPE primarily distinguished by their selection of conditional density estimators. Mixture Density Network (MDN)(also known as the mixture of Gaussians), being used as the estimator at the very beginning

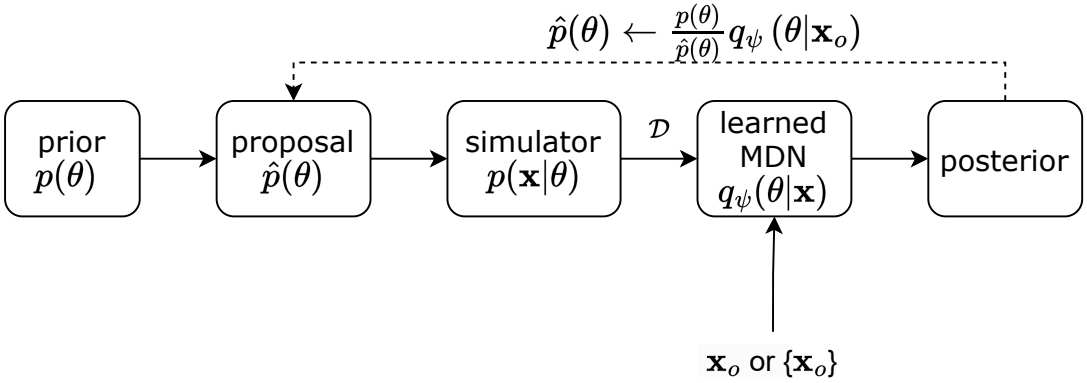


Figure 2.6: General scheme of SNPE-A proposed by [Papamakarios and Murray, 2016], here $\mathcal{D} = \{(\theta_i, \mathbf{x}_i)\}_{i=1}^N$ is the simulated data used to train MDN, algorithms can either directly use a prior to propose parameters or use sequential refinement of the proposal. Note that this figure ignore the case with summary statistics, in that case symbols should be changed to $S(\mathbf{x})$.

[Papamakarios and Murray, 2016, Lueckmann et al., 2017], and later on the gradual emergence of NPEs with Normalizing Flow [Greenberg et al., 2019, Winkler et al., 2019, Papamakarios et al., 2017] and Generative Adversarial Networks (GANs) [Pacchiardi and Dutta, 2022, Ramesh et al., 2022, Goodfellow et al., 2014] as estimators which have stronger representativeness than MDN for NPE. As we mentioned in the ABC subsection, to increase the acceptance rate, SMC-ABC and MCMC-ABC achieve this by modifying the way they propose the parameter (refining the proposal distribution iteratively). Similarly, NPE has a version called Sequential Neural Posterior Estimation (SNPE) for better simulation efficiency when we are only interested in the posterior for a single observation $\mathbf{x}_o$.

The idea behind SNPE is as follows: the standard NPE uses the prior $p(\theta)$ to propose parameters and then learns an estimator $q_\psi(\theta|\mathbf{x})$ for all $\mathbf{x}$. However, since we are only interested in the posterior at $\mathbf{x}_o$, simulations from parameters with very low posterior density $p(\theta|\mathbf{x}_o)$ are not particularly informative for learning ψ, as these simulations provide little information about $\theta|\mathbf{x}_o$. Thus, a significant portion of the simulation budget may be wasted on generating uninformative simulations, leading to low simulation efficiency.

Thus, to address this issue, SNPE modify the proposal distribution $\hat{p}(\theta)$ to be more informative about $\theta|\mathbf{x}_o$ during the training step. This is achieved through a sequential training procedure, where preliminary estimates of the posterior (obtained using the exact prior $p(\theta)$) are used as proposals for obtaining more training data. By iteratively refining the proposal distribution to better approximate the true posterior at $\mathbf{x}_o$, SNPE focuses the simulation budget on more informative regions of the parameter space, thereby improving simulation efficiency. We illustrate the workflow of (S)NPE in the Figure 2.6.

However, SNPE has a crucial issue is that with the use of proposal $\hat{p}(\theta)$ instead of prior $p(\theta)$ in training step: the final output becomes a biased approximation of the true posterior. This can be seen from the following equation:

$$q_\psi(\theta|\mathbf{x}_o) \propto p(\mathbf{x}_o|\theta)\hat{p}(\theta), \tag{2.19}$$

while our target posterior is $p(\mathbf{x}_o|\theta)\, p(\theta)$. To address this deviation from the true posterior, researchers have developed variants of SNPE, notably SNPE-A [Papamakarios and Murray, 2016], SNPE-B [Lueckmann et al., 2017], and SNPE-C [Greenberg et al., 2019].

The choice between NPE and SNPE primarily depends on the features of the data we are interested in. SNPE is particularly effective for single data points, offering enhanced simulation efficiency by focusing simulations more directly on the data point of interest. This makes it highly suitable for tasks that derive high-quality inferences from specific observations. On the other hand, NPE is better suited for scenarios involving multiple independent and identically distributed (i.i.d) observations. Its main advantage lies in amortized inference, which means that once trained, the model can quickly approximate posterior distributions for new observations without the need for re-training. This feature is highly beneficial for analyzing large datasets, where computational resources and efficiency are critical considerations [Lückmann, 2022].

2.3.2 Neural Likelihood Estimation

Unlike NPE, which directly estimates the posterior, Neural Likelihood Estimation (NLE) focuses on approximating the true likelihood $p(\mathbf{x}|\theta)$ with a surrogate likelihood $q_\psi(\mathbf{x}|\theta)$ using neural networks.

Methodology This approach involves generating a dataset $\mathcal{D} = \{(\theta_i, \mathbf{x}_i)\}_{i=1}^N$, where each sample $(\theta_i, \mathbf{x}_i)$ is drawn from the joint distribution $\hat{p}(\theta)p(\mathbf{x}|\theta)$ using the prior or proposal distribution $\hat{p}(\theta)$ and the simulator $p(\mathbf{x}|\theta)$. Given this dataset, a conditional density estimator $q_\psi(\mathbf{x}|\theta)$ is trained to serve as a surrogate likelihood by maximizing the total log-likelihood $\mathcal{L}(\psi) = \mathbb{E}_{\hat{p}(\theta)p(\mathbf{x}|\theta)} \log q_\psi(\mathbf{x}|\theta) \approx \frac{1}{N}\sum_{i=1}^N \log q_\psi\left(\mathbf{x}_i|\theta_i\right)$. After training, during inference time, given an observation $\mathbf{x}_o$, an estimate of the likelihood can be obtained by evaluating the surrogate at the observation point, $q_\psi(\mathbf{x} = \mathbf{x}_o|\theta)$. Finally, VI or MCMC can be employed to approximate the posterior at the observation point with this surrogate likelihood.

Related Work Similar to NPE, the field has seen a variety of approaches in choosing the parameterized surrogate likelihood, q_ψ, ranging from traditional synthetic likelihood

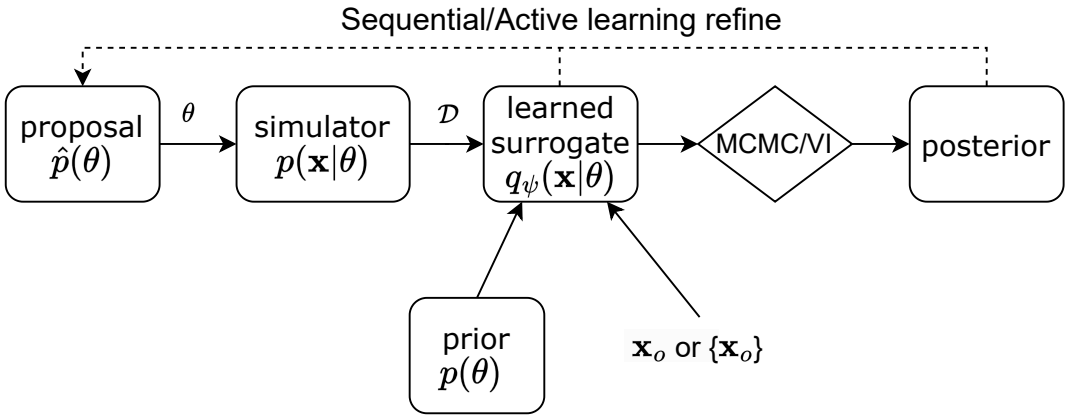


Figure 2.7: General scheme of NLE proposed by [Papamakarios, 2019, Lueckmann et al., 2019], here $\mathcal{D} = \{(\theta_i, \mathbf{x}_i)\}_{i=1}^N$ is the simulated data used to train neural networks-based surrogate, algorithms can either directly use a prior to propose parameters, use sequential refinement of the proposal or active learning on proposal or surrogate. And like the Figure 2.6, this figure also ignore the case with summary statistics, in that case symbols should be changed to $S(\mathbf{x})$.

methods employing multivariate normal distributions, as discussed by [Wood, 2010], to more recent advancements. Notably, [Papamakarios, 2019] utilized normalizing flows, and [Lueckmann et al., 2019] tailored the choice of surrogate families based on a deep understanding of specific simulators. Beyond the selection of surrogate families, like SNPE, there are also versions of NLE that aim to improve simulation efficiency; these methods are always called SNLE or SLE.

Two versions of SNLE are widely used. The first one is the SNLE proposed by Papamakarios [2019], which adopted a sequential scheme similar to SNPE-A. The idea is also proposing simulations with a proposal distribution, which is the posterior in the previous round. Another version of SLE was proposed by Lueckmann et al. [2019], who developed an active learning scheme to select new parameters. We illustrate the workflow of (S)NLE in the Figure 2.6

Comparison With NPE NLE holds several advantages over NPE. Firstly, NLE exhibits greater generality compared to NPE, which confines its flexibility to certain distribution families. For instance, SNPE-A can only be employed in conjunction with mixture of Gaussian estimators and exponential-family priors. In contrast, NLE can be utilized with any estimator and prior distribution, as there is no concern regarding issues arising from the proposal distribution (no need to correct for proposing parameter samples from a distribution other than the prior, as shown in Equation 2.19). Secondly, in the special case of inference with i.i.d observations $\mathbf{x_o}^1, \ldots, \mathbf{x_o}^N$ ($N > 1$), NLE is a better choice. As previously discussed, NPE directly approximates the true posterior with a density estimator $q_\psi(\theta|\mathbf{x})$, performing

well with a single observation. However, this method lacks a clear approach to modeling $q_\psi(\theta|\mathbf{x_o}^1, \ldots, \mathbf{x_o}^N)$. Consequently, for multiple observations, NPE must be trained with samples $(\theta_i, \mathbf{x}_s^1, \ldots, \mathbf{x}_s^N) \sim p(\theta) \prod_j p(\mathbf{x}_s^j|\theta)$ to obtain an amortized approximation of the target posterior. This also incurs a significant computational cost in simulation, as the simulator must be executed N times for each θ to obtain the complete simulation. Thus, sample efficiency is reduced, especially when N is large. However, NLE circumvents this issue as it directly models the likelihood function $q_\psi(\mathbf{x}|\theta)$, which consequently models the posterior $q_\psi(\theta|\mathbf{x_o}^1, \ldots, \mathbf{x_o}^N)$ clearly. For multiple i.i.d observations, the surrogate becomes:

$$p(\theta) \prod_{i=1}^{N} q_\psi(\mathbf{x_o}^i|\theta) \tag{2.20}$$

which is always tractable with the learned $q_\psi(\mathbf{x}|\theta)$. Thus, because of this feature of NLE (analytically factorizing $q_\psi(\theta|\mathbf{x_o}^1, \ldots, \mathbf{x_o}^N)$), during training, it only requires single-observation/parameter pairs, thereby maintaining sample efficiency. Nonetheless, NLE possesses an obvious drawback. As outlined above, in contrast to NPE, which directly models the posterior, NLE necessitates an additional step during inference to generate posterior samples with MCMC or VI. This step can be computationally expensive or inefficient for complex posteriors.

2.3.3 Neural Ratio Estimation

Overview The motivation behind Neural Ratio Estimation (NRE) stems from the challenges encountered by MCMC samplers when dealing with intractable likelihoods. MCMC methods, such as the Metropolis-Hastings algorithm (for more details, see [Brooks et al., 2011]), aim to generate samples from the target posterior distribution $p(\theta|\mathbf{x})$ by constructing a Markov chain that explores the parameter space. At each iteration, given the current state θ_t, a new candidate parameter θ' is proposed from a proposal distribution $q(\theta'|\theta_t)$. The acceptance probability α for transitioning from θ_t to θ' is computed as:

$$\alpha(\theta_t, \theta') = \min\left\{1, \frac{p(\theta')p(\mathbf{x}|\theta')q(\theta_t|\theta')}{p(\theta_t)p(\mathbf{x}|\theta_t)q(\theta'|\theta_t)}\right\}. \tag{2.21}$$

In scenarios where the likelihood function $p(\mathbf{x}|\theta)$ is intractable, the evaluation of the likelihood ratio $\frac{p(\mathbf{x}|\theta')}{p(\mathbf{x}|\theta_t)}$ becomes infeasible, rendering MCMC methods ineffective. To overcome this limitation, NRE aims to model the intractable likelihood ratio directly using an amortized ratio predictor $r(\mathbf{x}|\theta', \theta_t) = \frac{p(\mathbf{x}|\theta')}{p(\mathbf{x}|\theta_t)}$, enabling MCMC methods to function as intended, even in the presence of intractable likelihoods. For more details on related methods, see [Hermans, 2022, Lueckmann et al., 2021, Hermans et al., 2020].

3. Related Work

In this chapter, we review related literature, focusing on existing works regarding neural point estimators and SBI methods in a BDM context.

3.1 Neural Point Estimator

The concept of using neural networks for amortized statistical inference, particularly for point estimation, was initially introduced by Chon and Cohen [1997]. They trained feedforward neural networks (FNNs) to estimate autoregressive moving average model parameters, demonstrating that neural networks could approximate true statistical values, termed Bayes estimators, with high accuracy. Despite its innovative approach, this method didn't immediately spark widespread research into neural network-based point estimation.

It was not until more recent advancements that the application of neural networks for point estimation began to gain widespread recognition. Creel [2017] demonstrated that FNNs could not only directly estimate model parameters but also augment classical or Bayesian inference methods, particularly in financial modeling contexts. This versatility highlighted neural networks' adaptability across various inference tasks.

A significant advancement was made by Chan et al. [2018], who applied a permutation-invariant neural architecture to population genetics, overcoming the limitations associated with traditional ABC methods' reliance on hand-crafted summary statistics. This approach enhanced inference accuracy by allowing neural networks to directly process complex data sets. Further contributions include Banesh et al. [2021], who utilized convolutional neural networks (CNNs) for efficient and accurate parameter estimation in spatial data modeling, outperforming traditional Maximum Likelihood Estimation (MLE) methods in both speed and predictive performance. Similarly, Rudi et al. [2022]'s work on time-series data analysis confirmed the feasibility and efficiency of neural network-based estimators for dynamic data sets.

These studies underscore neural networks' potential in statistical inference, especially for complex simulation tasks, paving the way for future research in this promising field.

3.2 SBI-based Bayesian Decision Making

Bayesian Decision Making (BDM) is a systematic approach that integrates Bayesian inference principles to facilitate decision-making in the under uncertainty. At its core, BDM involves a prior $p(\theta)$ about an unknown parameter θ, a likelihood $p(\mathbf{x}|\theta)$, and a cost function $c(\theta, a)$ quantifies the consequences of taking a particular action a if the true parameters θ were known. The optimal action, a^*, is defined mathematically as:

$$\begin{aligned} a^* &= \arg\min_{a \in A} \int c(\theta, a) \cdot p(\theta|\mathbf{x}) d\theta \\ &= \arg\min_{a \in A} \mathbb{E}_{p(\theta|\mathbf{x})}[c(\theta, a)], \end{aligned} \tag{3.1}$$

where $\mathbb{E}_{p(\theta|\mathbf{x})}[L(\theta, a)]$ denotes the expected loss over the posterior of θ given the data $\mathbf{x}$.

[Gorecki et al., 2023] introduced a method called Bayesian Amortized Decision Making (BAM), which streamlines Bayesian decision-making in likelihood-free contexts by directly estimating the expected cost. Unlike other neural SBI methods used in BDM, which first approximate the full posterior and then utilize methods like Monte Carlo to estimate the expected cost for actions a:

$$\mathbb{E}_{p(\boldsymbol{\theta}|\mathbf{x}_o)}[c(\boldsymbol{\theta}, \mathbf{a})] \approx \frac{1}{N} \sum_{i=1}^{N} c(\boldsymbol{\theta}_i, \mathbf{a}) \tag{3.2}$$

where N is the number of simulated data, BAM optimizes a feedforward neural network to predict the expected action cost based on simulated data from the joint distribution $p(\theta, \mathbf{x})$. During training, the loss function is the mean square error between the predicted cost and the true cost:

$$\mathcal{L}(\phi) = \frac{1}{|N|} \left(f_\phi(\mathbf{x}, \mathbf{a}) - c(\boldsymbol{\theta}, \mathbf{a}) \right)^2, \tag{3.3}$$

where ϕ is the parameters of the network.

Thus, after training, given a data-action pair $(\mathbf{x}, \mathbf{a})$, the network can output the approximate expected cost because minimizing the MSE is equivalent to approximating the mean value of the target.

At decision-making time, given observed data $\mathbf{x}_o$ and a set of actions, the best action is obtained by finding the minimal expected cost. Experimental results indicate BAM's simplicity and efficiency over other neural SBI methods like NPE-MC, which requires posterior approximation and subsequent Monte Carlo estimation for each observation. This research's core idea is very similar to ours, and it opens up a different view for us to treat our work as a downstream task of BDM and apply our work on more general BDM, which could be a potential future work of our research.

4. Neural Amortization of Bayesian Point Estimation

In scenarios where likelihood functions are intractable, the conventional approach involves employing SBI methods to infer the full posterior distribution before calculating point estimations. The good thing is that we can do a comprehensive Bayesian analysis with a full posterior. However, for some downstream work, such as point estimation, these conventional methods have to rerun analytical computations or sampling for each observation, making point estimating computationally intensive and not fully amortized, which makes this task inefficient. Moreover, modeling point estimation as a downstream task of BDM reveals that reliance on approximate posteriors could lead to suboptimal point estimations.

To address these challenges, we introduce the Neural Amortization of Bayesian Point Estimation (NBPE) method. This approach directly obtains accurate, desired point estimations, particularly suited for simulator models with inherently intractable likelihoods.

4.1 Methodology

Our methodology operates within the framework of Neural Network-based SBI. Central to our approach is the division of the neural network architecture into two principal components: the summary network $\mathcal{H}_\phi(\mathbf{x})$ for extracting summary statistics $S(\mathbf{x})$ from the data $\mathbf{x}$, and the inference network $\mathcal{I}_\psi(S(\mathbf{x}), \alpha)$ for deducing point estimates given the summary statistics and a specified value of α.

Given a simulator model characterized by a prior distribution $p(\theta)$ and a likelihood function $p(\mathbf{x}|\ \theta)$, we begin by generating a dataset $\mathcal{D} = \{(\theta_i, \mathbf{x}_i)\}_{i=1}^N$ drawn from the joint distribution $p(\theta, \mathbf{x})$. The dataset serves as the training input for both networks.

To facilitate the co-training of the summary and inference networks, we optimize a joint loss function $L(\phi, \psi)$, defined as:

$$L(\phi, \psi) = \frac{1}{N} \sum_{i=1}^{N} \left|\theta_i - \mathcal{I}_\psi \left(S\left(\mathbf{x}_i\right), \alpha\right)\right|^\alpha \tag{4.1}$$

where θ_i represents parameters sampled from $p(\theta)$, and $S(\mathbf{x}_i) = \mathcal{H}_\phi(\mathbf{x}_i)$ signifies the summary statistics generated for the i-th data point. This collaborative training approach ensures that the loss function guides both the summary extraction and inference prediction processes, with α playing a crucial role in determining the focus of the estimation task.

During inference, for a given observation x_o, by setting the value of α, the inference network I_ψ quickly provides the desired posterior statistic, be it the mean, median, or mode. This approach ensures that our network is not merely a generic feedforward neural network but a specialized architecture tailored for efficient and precise point estimation within the SBI context:

Proposition 1 *Let $p(\theta, \mathbf{x})$ be the joint distribution of parameters θ and data $\mathbf{x}$. The summary network $\mathcal{H}_\phi(\mathbf{x})$ outputs summary statistics $S(\mathbf{x})$. The inference function $\mathcal{I}_\psi(S, \alpha)$ estimates θ using $S(\mathbf{x})$ and a specified α. It is optimized by minimizing the powered absolute error loss:*

$$L(\phi, \psi) = \frac{1}{N} \sum_{i=1}^{N} \left|\theta_i - \mathcal{I}_\psi\left(S(\mathbf{x}_i), \alpha\right)\right|^\alpha.$$

The choice of α targets different statistics of the posterior $p(\theta|\mathbf{x})$:

- $\alpha = 2$ *targets the mean*
- $\alpha = 1$ *targets the median*
- $\alpha \to 0$ *targets the mode*

Proof provided in Appendix A.

Our method's integration into Bayesian Decision Making frameworks further exemplifies its novelty. BDM traditionally seeks the optimal action a^*, yet in our context, decision-making converges on identifying the best estimate that aligns closely with real statistics, transforming the action space into the parameter space Θ. Consequently, the cost function in BDM becomes:

$$c(\theta, \hat{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \left|\theta_i - \hat{\theta}\right|^\alpha, \tag{4.2}$$

with $\hat{\theta}$ as our predictions, reshapes the goal into finding the best estimate θ^* by minimizing the expected loss:

$$\theta^* = \arg\min_{\theta \in \Theta} \mathbb{E}_{p(\theta|\mathbf{x})}[c(\theta, \hat{\theta})]. \tag{4.3}$$

This optimization aligns with our method, predicting the optimal θ using a parameterized neural network. By conceptualizing NBPE as a downstream BDM task, we

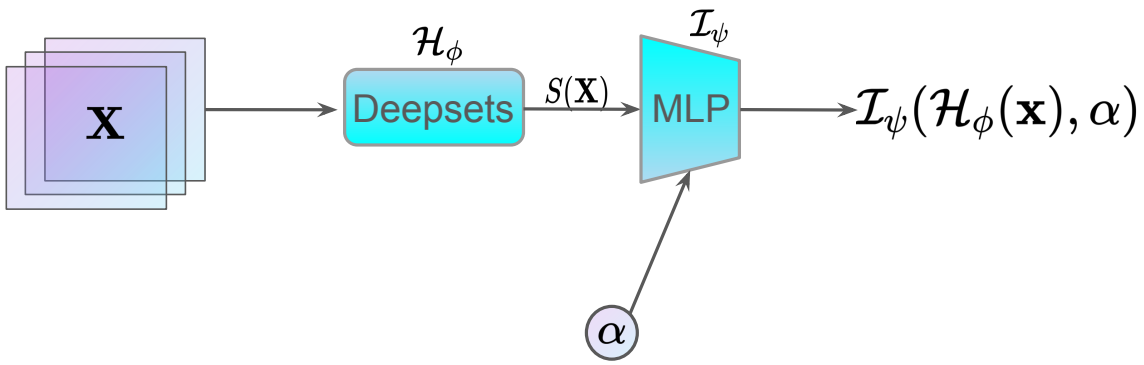


Figure 4.1: Overview of the NBPE network architecture, highlighting the two-component structure for Bayesian point estimation. The input set X is initially transformed by the summary network H_ϕ, typically implemented using Deepsets, to generate a summarized representation $S(X)$. This summary, along with a specified exponent α, guides the inference network I_ψ, embodied by an MLP, to produce the Bayesian point estimate $I_\psi\left(H_\phi(X), \alpha\right)$. The system allows for the substitution of Deepsets with SetTransformer, offering flexibility in the choice of summary statistic extractor based on the complexity and nature of the dataset.

.

indirectly address the limitations of classical SBI approaches in BDM applications, such as computational inefficiency and suboptimal outcomes due to approximate posteriors.

In summary, NBPE leverages the foundational theory underpinning SBI but distinguishes itself by training an end-to-end neural network-based estimator. This approach not only achieves preferred point estimation efficiently but, by theoretically modeling it as a Bayesian decision-making task, also demonstrates its potential to circumvent challenges faced by traditional NN-SBI methods in BDM contexts.

4.2 Networks Architecture

The network architecture employed in the NBPE method is fundamentally an encoder-decoder structure, a paradigm widely recognized for its efficacy in various domains, notably in natural language processing (NLP) applications as demonstrated by models such as BART [Lewis et al., 2020] and the Transformer [Vaswani et al., 2017]. In the context of NBPE, the encoder $\mathcal{H}_\phi$'s role is to extract informative summary statistics ($S(\mathbf{x})$) from the input data ($\mathbf{x}$), capturing the essence of the data in a compressed form that retains its critical statistical properties. This process is crucial for ensuring that the neural network focuses on the most relevant aspects of the data for the task at hand and hugely reduce computational cost.

For the encoder, a simplified version of Deepsets is employed(we already illustrated it in Figure 2.5), primarily due to its permutation-invariant characteristics. This choice is dictated by the nature of the training data, which, being simulation

outputs, can often be considered as sets where the order of elements is not fixed. The permutation invariance of Deepsets ensures that the model's output remains consistent regardless of the input's order, a necessary property for dealing with set data effectively. While Deepset serves as the primary encoder, offering a balance of performance and computational efficiency, the SetTransformer is introduced as an alternative with the potential for higher complexity and representational capacity at the cost of increased computational demands.

For the decoder $\mathcal{I}_\psi$, the architecture integrates an MLP, tasked with transforming the summary statistics back into the domain of interest, namely the Bayesian point estimates. This step is crucial for converting the abstracted statistical information back into actionable insights. The whole structure is outlined in the Figure 4.1.

4.3 Training Methodology

Our training methodology is engineered to facilitate point estimation for different statistical quantities during inference by adjusting the value of α. To this end, we have constructed a network that incorporates α as a modifiable parameter, allowing users to specify their preferred statistical measure by setting α within a predetermined range or from a specified discrete set.

Our approach involves two strategies for determining α. The first strategy defines a continuous range, such as $\alpha \in [1/4, 2]$, from which values are sampled uniformly in each training epoch. Alternatively, the second strategy employs a discrete set of predefined α values, for example, $\{1/4, 1/2, 1, 2\}$. In both cases, the chosen α is introduced into the decoding network as a constant vector, augmenting the encoded summary statistics with an additional dimension that reflects the desired statistical measure. This modified input is then processed by the decoder to produce the final estimation. For convenience, we denote the range of α as $\mathcal{A}$ for both continuous and discrete cases.

Another consideration for ensuring the robustness of our NBPE model is its ability to handle observations of varying sizes, which may differ from those presented during training. To address this, we introduce a range of possible observation sizes, specified by the hyperparameters $\mathcal{O}_{\min}$ and $\mathcal{O}_{\max}$. Within each epoch, we randomly select a simulation size (we denote it as $\mathcal{B}$) from this defined range to serve as the training data size for that epoch. This randomized approach to simulation size not only enhances the model's adaptability to different observational data sizes but also enriches its learning experience, promoting better generalization for diverse datasets. The detailed algorithm is outlined in Algorithm 2.

Algorithm 2 Neural Amortization of Bayesian Point Estimates (NBPE)

Input: Simulator $p(\mathbf{x}|\theta)$, prior $p(\theta)$, summary network $\mathcal{H}_\phi$, inference network $\mathcal{I}_\psi$, simulation budget $\mathcal{N}$, range of α $\mathcal{A}$, observation size range $[\mathcal{O}_{\min}, \mathcal{O}_{\max}]$, training epochs E, i.i.d observations $\mathbf{X}_o$, user-specified alpha α_u.
Initialize training dataset $\mathcal{D} = \emptyset$.
for $i = 1$ to $\mathcal{N}$ **do**
    Sample parameter from prior: $\theta_i \sim p(\theta)$.
    Generate synthetic data using simulator: $\mathbf{x}_i \sim p(\mathbf{x}|\theta_i)$.
    Append $(\theta_i, \mathbf{x}_i)$ pair to training dataset: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\theta_i, \mathbf{x}_i)\}$.
end for
for $j = 1$ to E **do**
    Sample a simulation size $\mathcal{B}_j$ uniformly from $[\mathcal{O}_{\min}, \mathcal{O}_{\max}]$.
    Sample an α_j for the current epoch: $\alpha_j \sim \mathcal{A}$.
    Select a batch $\{(\theta_b, \mathbf{x}_b)\}_{b=1}^{\mathcal{B}_j}$ randomly from $\mathcal{D}$.
    Compute loss for the batch: $L(\phi, \psi) = \frac{1}{\mathcal{B}_j} \sum_{b=1}^{\mathcal{B}_j} |\theta_b - \mathcal{I}_\psi(\mathcal{H}_\phi(\mathbf{x}_b), \alpha_j)|^{\alpha_j}$.
    Update ϕ and ψ parameters using backpropagation.
end for
return Trained summary network $\mathcal{H}_\phi$, inference network $\mathcal{I}_\psi$, and desired posterior statistics $\theta^* = \mathcal{I}_\psi(\mathcal{H}_\phi(\mathbf{X}_o), \alpha_u)$.

5. Experiments

This chapter demonstrates the ability of our proposed method to perform accurate and flexible point estimation by selecting the value of the power of the loss function α, during inference time. We conduct experiments on three different models to assess the performance of our approach.

First, we compare the accuracy and efficiency of our method with another amortized inference method, NPE, from the BayesFlow SBI software, using a toy 4D Gaussian model. This comparison serves as a basic validation of our method's capability to produce accurate and fast point estimates.

Next, we further demonstrate this capability on a more complex model, the Bernoulli-Gaussian Linear Model (GLM), which is a typical simulator model with an implicitly defined likelihood. This experiment aims to showcase the competitiveness of our approach compared to existing neural methods on complex simulator models.

Finally, we demonstrate the flexible point estimation feature of our method using a Poisson-Gamma model. By measuring the change in distance from the true mean, median, and mode as α varies, we illustrate how our method can effectively target different desired statistics of the posterior distribution. Additional experiments and brief analyses for flexible point estimation are provided in Appendix A.4 as supplementary material, as their results are somewhat redundant, and hence, we have moved them to the appendix.

5.1 Performance Metrics

In the experiments we will use the following metrics to measure the performances: R^2 score and mean squared error(MSE).

5.1.1 R^2 Score

To measure the accuracy of point estimations, that is, the capacity to retrieve the model's "true" parameters, we propose using the R^2 score, commonly known as the coefficient of determination.

The R^2 score is a statistical measure of the strength of the linear relationship between two variables. In the context of model performance, it measures how well the predictions of a regression model correspond to the true parameters. The R^2 score is defined as:

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(\theta_i - \theta_i^*)^2}{\sum_{i=1}^{N}(\theta_i - \bar{\theta})^2} \tag{5.1}$$

where N is the number of parameters, θ_i represents the true parameter values, θ_i^* denotes the estimated values from the model, and $\bar{\theta}$ is the mean of the true parameter values.

An R^2 score of 1 means the model's predictions perfectly match the true parameters. In contrast, an R^2 score of 0 means the model's predictions do not correspond to the true parameters, performing no better than simply predicting the mean of the true parameters for all inputs. Negative R^2 values indicate that the model performs worse than predicting the mean. This score provides an easy-to-understand measure of a model's accuracy, with higher values indicating better performance.

In our experiments, we compare the R^2 scores of our NBPE with those of MAP and BayesFlow estimations. The R^2 score allows us to get a straightforward way to see how well our method can match or even outperform other methods in recovering true parameters.

5.1.2 Mean squared error

Mean Squared Error (MSE) is a widely used metric for quantifying the accuracy of estimators in statistical models. In the context of our NBPE method, MSE is defined as:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}\left(\theta_i^* - \theta_i\right)^2.$$

MSE is suitable for our study as it is sensitive to large errors. This characteristic is particularly relevant when evaluating the performance of NBPE across different values of the exponent α, as it can accurately reflect the divergence of the estimated point from the true parameter value.

5.2 Performance Evaluation of Parameter Recovery

Accurately and rapidly recovering true data-generating parameters is a crucial test for any parameter estimation method. It demonstrates the method's ability to produce

accurate and useful approximations to serve as reliable inputs for downstream analyses and applications. If a parameter cannot be recovered with reasonable precision, we cannot trust its estimate, making it unsuitable for further utilization. To thoroughly evaluate the parameter recovery capabilities of our NBPE method, we conducted experiments on two different types of statistical models and compared the performance of these models with the NPE method from the BayesFlow SBI software; the first model is a toy 4D Multivariate Gaussian model and the second one is a complex simulator model called Bernoulli-Generalized Linear Model (Ber-GLM).

5.2.1 Toy 4D Multivariate Gaussian model

We begin our empirical investigation with a relatively simple 4D Gaussian model, a widely used toy example in parameter estimation. This initial experiment validates our method's fundamental performance on a tractable problem where the true parameter values are available, enabling precise accuracy assessment. To demonstrate the power of our method, we also applied NPE to approximate the posterior with neural posterior estimation and learned summary statistics [Radev et al., 2020], as implemented in the BayesFlow software for amortized Bayesian workflows [Radev et al., 2023]. After we get the approximate posterior, we compute the posterior mean by the Monte Carlo method to recover true parameters and then compare it with the results of our method, which end-to-end recovers parameters without the full posterior.

Model Description The 4D Gaussian model under consideration takes the following form:

$$\begin{aligned} \boldsymbol{\mu} &\sim \mathcal{N}_D(\mathbf{0}, \sigma_0 \mathbb{I}) \\ \mathbf{x} &\sim \mathcal{N}_D(\boldsymbol{\mu}, \sigma_1 \mathbb{I}) \end{aligned} \tag{5.2}$$

where $\mathcal{N}_D$ denotes a multivariate Gaussian density with D dimensions, which we set at $D = 4$ for the current example. For simplicity, we will also set $\sigma_0 = 1$ and $\sigma_1 = 1$.

Set up Our neural architecture employs DeepSets as the summary network, processing tensors of shape (N_{sim}, N_{obs}, D), where N_{sim} is the number of simulations, N_{obs} is the number of simulated data points per proposed parameter, and D is the data dimensionality. The output of the summary network is 2D tensors of shape (N_{sim}, D_{summary}), where $D_{\text{summary}} = 10$ is a hyperparameter that should not be lower than the number of parameters in the model, as suggested by Radev et al. [2020]. A Multi-Layer Perceptron (MLP) with two hidden layers of 32 neurons each serves as our inference network. Since we are only interested in the ability to recover parameters, we take a fixed $\alpha = 2$ (targetting posterior mean) for training and evaluation. In NPE, the

invariant network is used as the summary network, and the conditional Invertible Neural Network (cINN) is used as the inference network (for more details about these two networks, see [Radev et al., 2023]). The D_{summary} is also 10. We use the Monte Carlo method to get the posterior mean with the approximate posterior.

For training NBPE, we use the default Adam optimizer, set the epoch count at 5000, and the simulation batch size per epoch at 32. Since we aim to demonstrate the accuracy and efficiency of our model in recovering parameters, we set the power of the loss $\alpha = 2$ (we do not use this experiment to show that by changing α, we can approximate different statistics of the posterior, as the Gaussian model's mean, median, and mode are identical). The range of observation size [$\mathcal{O}$min, $\mathcal{O}$max] is set between 1 and 50. For training NPE, we use the default training method implemented in BayesFlow, typically Adam optimizer with a learning rate of 5e-4 and a cosine learning rate decay from 5e-4 to 0. We also set the epoch count at 5000 and the simulation batch size per epoch at 32 to keep the simulation budget the same as ours. And The range of observation size [$\mathcal{O}$min, $\mathcal{O}$max] is also between 1 and 50.

For evaluation, we conduct two sets of experiments. In the first set, we assess the inference accuracy of NBPE by comparing it with the NPE. We generate a dataset with a shape of (1000, 50, 4), where 1000 represents the batch size or simulation size, 50 is the fixed observation size, and 4 is the dimensionality of the 4D data. We then feed this dataset into both NBPE and NPE methods and compute the R^2 score and MSE between the estimated parameters and the true parameters to evaluate the accuracy performance. To ensure robust and reliable results, we repeat this process 100 times and report the mean R^2 score, mean MSE, and their respective standard deviations.

In the second set of experiments, we investigate the computational efficiency of NBPE in terms of inference speed. We first generate a large dataset with a shape of (10000, 50, 4) from the simulator. Then we perform three tests on this dataset, each with a different batch size: 100, 1000, and 10000. For each batch size, we measure the time required for inference using both NBPE and NPE methods. Consistent with the inference accuracy evaluation, we repeat this process 100 times to obtain stable and reliable results, and report the mean inference time and its standard deviation for each batch size and method.

Results Analysis Table 5.1 and Figure 5.1 illustrate that NBPE achieves a very high R^2 score (>0.9) and a very small value of MSE on this toy model, indicating its ability to successfully recover the majority of the true data-generating parameters. Moreover, upon comparing with the results of NPE, we observe that NPE can achieve slightly better performance than NBPE. In our opinion, this can be attributed to the stronger network architecture employed in NPE, which tends to approximate the true posterior

more accurately in this particular toy model, leading to more precise corresponding point estimates. However, Table 5.2 clearly shows our method is way more efficient than NPE: on each different size of the dataset, NBPE can perform approximately 200 times faster than NPE. This observation is valuable since it demonstrates the advantage of our end-to-end method; we can save the inference time hugely by skipping the extra time-consuming step (compute the point estimation) and just lose a small quantity of accuracy.

In summary, this experiment demonstrates that NBPE can attain competitive accuracy in point estimation compared to NPE while being significantly more efficient.

	Metrics	
	R^2	MSE
NBPE	0.964±0.021	0.035±0.002
NPE	0.975±0.017	0.025±0.001

Table 5.1: Comparison of inference accuracy metrics (R^2 score and Mean Squared Error) between NBPE and NPE methods on the toy 4D Gaussian model, with results produced from 100 repetitions reporting the mean and standard deviation.

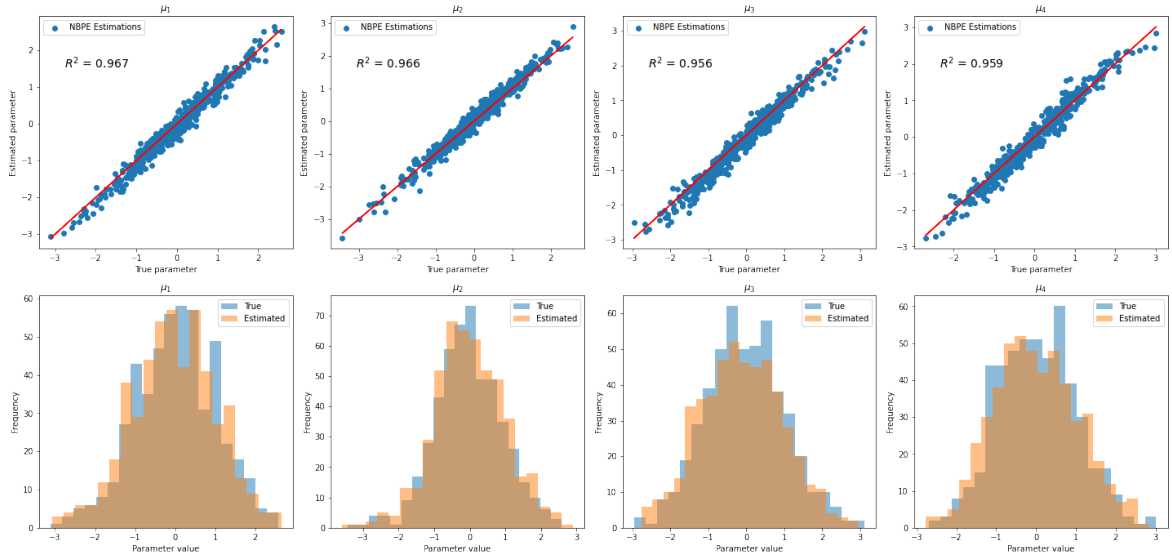
Batch Size	NPE	NBPE
100	177.609±32.957 ms	0.802±0.408 ms
1000	821.872±78.599 ms	3.496±1.236 ms
10000	8599.183±129.594ms	54.339±8.764 ms

Table 5.2: Comparison of mean inference time in milliseconds and standard deviation for different batch sizes between NPE and NBPE methods on the 4D toy Gaussian model, obtained from 100 repetitions.

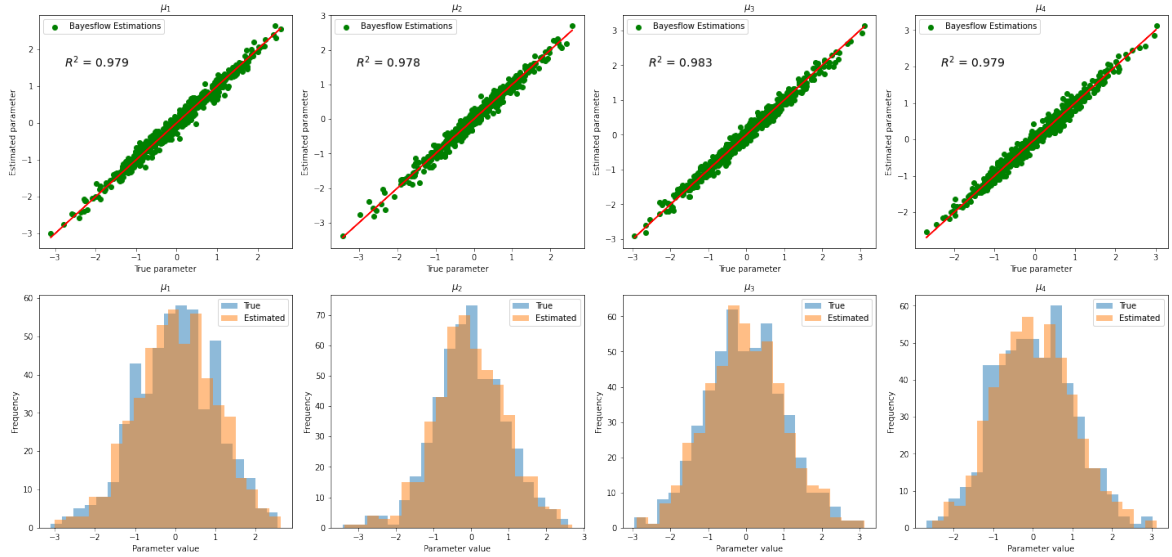
5.2.2 Bernoulli-Generalized Linear Model

In this experiment, we evaluate our proposed NBPE method on a simulator model to assess its ability to accurately recover parameters. We begin with a relatively straightforward simulator model, where the simulating process is not overly expensive, and the model parameters are relatively easier to infer compared to other simulator models.

Model description The model we are working on is a 10-parameter Generalized linear model (GLM) with Bernoulli observations and Gaussian prior with a covariance matrix, which encourages smoothness by penalizing the second-order differences in the



(a)



(b)

Figure 5.1: Comparison of parameter recovery using NPE and NBPE (our method) for a 4D Gaussian model. (a) shows the results using the NBPE, our proposed method, while (b) presents results from the NPE. Each column corresponds to one of the four parameters of the Gaussian model. The top row of each subplot illustrates scatter plots of the estimated versus true parameter values, with the coefficient of determination $\left(R^2\right)$ provided for each parameter. The bottom row shows histograms comparing the distributions of parameter estimates. In these histograms, the true parameter values are labelled as 'True', and the estimations from NBPE or NPE are labelled as 'Estimated'.

vector of the parameter [De Nicolao et al., 1997]. Mathematically, the prior of this model could be expressed as:

$$\begin{aligned} \beta &\sim \mathcal{N}(0, 2) \\ \mathbf{f} &\sim \mathcal{N}\left(\mathbf{0}, \left(\mathbf{F}^\top \mathbf{F}\right)^{-1}\right), \end{aligned} \tag{5.3}$$

where $\mathbf{F}$ is defined such that $\mathbf{F}_{i,i-2} = 1, \mathbf{F}_{i,i-1} = -2, \mathbf{F}_{i,i} = 1 + \sqrt{\frac{i-1}{9}}$, and $\mathbf{F}_{i,j} = 0$ for $j \neq i-2, i-1, i$, with $1 \leq i, j \leq 9$. Since the expression of the simulator for this model is overcomplicated, we are not showing the simulator here. We recommend to read [Lueckmann et al., 2021] for more technical details about this model.

Set up Our neural architecture here also employs DeepSets as the summary network, the summary dimentionality D_{summary} here is 32. A MLP with three hidden layers of 16 neurons each serves as our inference network. We also take a fixed $\alpha = 2$ (targetting posterior mean) for training and evaluation. In NPE, the invariant network is used as the summary network, and the cINN is used as the inference network. The D_{summary} is also 32. We use the Monte Carlo method to get the posterior mean with the approximate posterior.

For training NBPE, we use the default Adam optimizer, set the epoch count at 10000, and the simulation batch size per epoch at 32. The range of observation size [$\mathcal{O}$min, $\mathcal{O}$max] is set between 1 and 15. For training NPE, we use the same setting for the Adam optimizer as our privous Gaussian experiment. We also set the epoch count at 10000 and the simulation batch size per epoch at 32 to keep the simulation budget the same as ours. The range of observation size [$\mathcal{O}$min, $\mathcal{O}$max] is also between 1 and 15.

For evaluation, we also conduct two sets of experiments. In the first set, we assess the inference accuracy of NBPE by comparing it with the NPE. We generate a dataset with a shape of (1000, 15, 10). We then feed this dataset into both NBPE and NPE methods and compute the R^2 score and MSE between the estimated parameters and the true parameters to evaluate the accuracy performance. To ensure robust and reliable results, we repeat this process 100 times and report the mean R^2 score, mean MSE, and their respective standard deviations.

In the second set of experiments, we investigate the computational efficiency of NBPE in terms of inference speed. We first generate a large dataset with a shape of (10000, 15, 10) from the simulator. Then we perform three tests on this dataset, each with a different batch size: 100, 1000, and 10000. For each batch size, we measure the time required for inference using both NBPE and NPE methods. Consistent with the inference accuracy evaluation, we repeat this process 100 times to obtain stable and

reliable results, and report the mean inference time and its standard deviation for each batch size and method.

Result Analysis Similar to the previous experiment, Table 5.3 and Figure 5.2 show that NBPE achieves a competitive parameter recovery performance compared to NPE, with a slightly lower R^2 score and higher value of MSE. More interestingly, Table 5.4 clearly demonstrates NBPE's superior efficiency, being approximately 200 times faster than NPE on average across different dataset sizes. This experiment further demonstrates that NBPE can provide accurate point estimation while offering substantial computational efficiency gains.

	Metrics	
	R^2	MSE
NBPE	0.909±0.049	0.149±0.010
NPE	0.947±0.033	0.087±0.006

Table 5.3: Comparison of inference accuracy metrics (R^2 score and Mean Squared Error) between NBPE and NPE methods on Bernoulli GLM, with results produced from 100 repetitions reporting the mean and standard deviation.

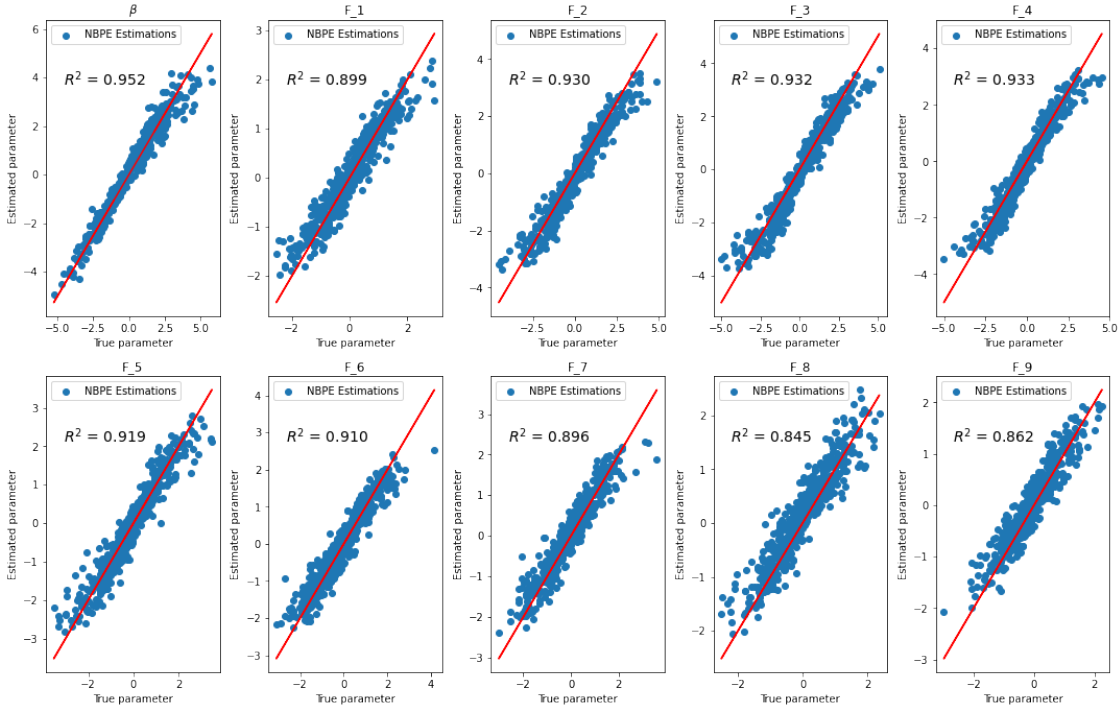
Batch Size	NPE	NBPE
100	170.402±42.578 ms	0.389±0.290 ms
1000	238.367±27.961 ms	1.679±1.959 ms
10000	1121.802±67.317 ms	5.341±1.092 ms

Table 5.4: Comparison of mean inference time in milliseconds and standard deviation for different batch sizes between NPE and NBPE methods on the Bernoulli GLM, obtained from 100 repetitions.

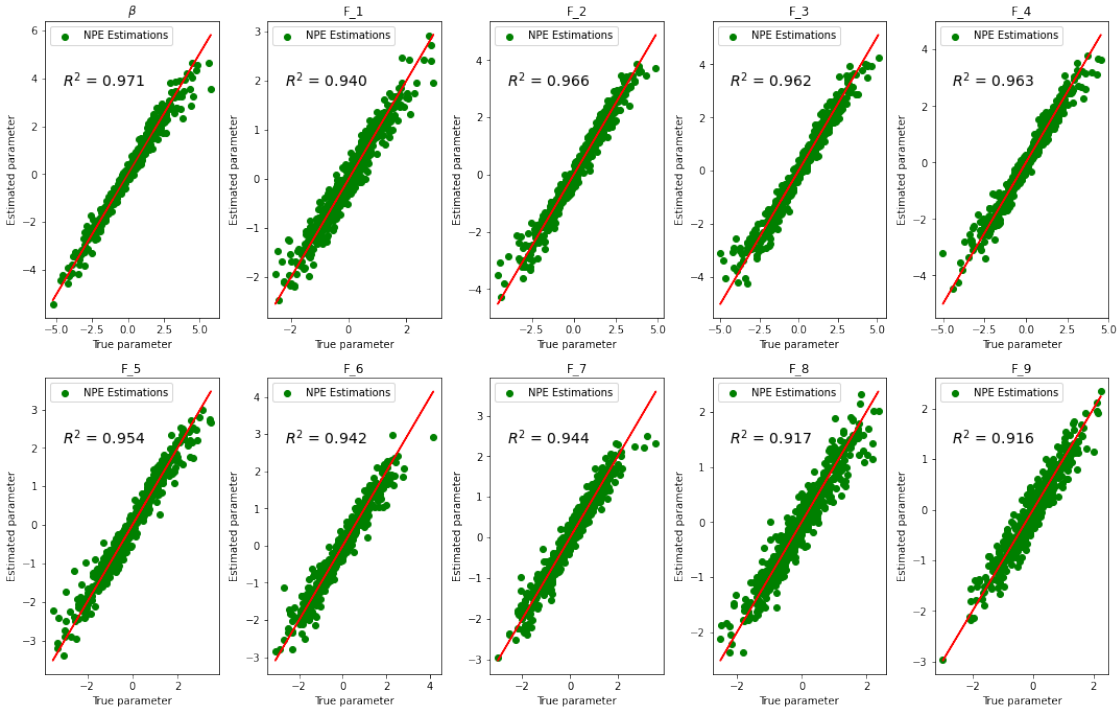
In summary, through experiments on a toy 4D Gaussian model and the Bernoulli GLM simulator model, we have shown NBPE's ability to perform accurate point estimation competitive with NPE, while significantly outperforming NPE in terms of inference time efficiency.

5.3 Performance Evaluation of Flexible Point Estimation

After testing the accuracy of the point estimations outputted by our method, we further demonstrate that our method can output different statistics of the target posterior distribution.



(a)



(b)

Figure 5.2: Comparison of parameter recovery using NPE and NBPE (our method) for a Bernoulli-GLM. Scatter plots show the relationship between true and estimated parameter values, with the R^2 score indicating the accuracy of recovery.

5.3.1 Experiment on Poisson-gamma model

Model description Formally, the model can be expressed as follows:

$$\begin{aligned} \lambda &\sim \text{Gamma}(2, 5) \\ \mathbf{x} &\sim \text{Poisson}(\lambda) \end{aligned} \tag{5.4}$$

Here, the Gamma distribution with a shape parameter $\alpha = 2$ and rate parameter $\beta = 5$ provides the prior for the rate λ of the Poisson distribution.

The Poisson-Gamma model is a conjugate pair, meaning that the posterior distribution is also a Gamma distribution. This makes calculations of posterior mean, median, and mode relatively straightforward under this Bayesian framework. Given the prior Gamma distribution parameters $\alpha = 2$ and $\beta = 5$, and the likelihood coming from a Poisson distribution, the posterior parameters for λ after observing the data x will be updated as follows:

$$\alpha_{\text{post}} = \alpha_{\text{prior}} + \sum_{i=1}^{N} \mathbf{x}_i, \ \beta_{\text{post}} = \beta_{\text{prior}} + N \tag{5.5}$$

where $\sum_{i=1}^{N} \mathbf{x}_i$ is the total count of events observed and N is the number of observations. The posterior mean of a Gamma distribution is given by $\frac{\alpha}{\beta}$, and thus, the posterior mean for λ is:

$$\text{Mean}_{\text{post}} = \frac{\alpha_{\text{post}}}{\beta_{\text{post}}} \tag{5.6}$$

The mode of a Gamma distribution with shape parameter k and scale parameter θ (or rate parameter $\beta = 1/\theta$) is $(k-1)\theta$ when k is greater than 1 , so the posterior mode is:

$$\text{Mode}_{\text{post}} = \frac{\alpha_{\text{post}} - 1}{\beta_{\text{post}}} \tag{5.7}$$

For the median, there is no closed-form expression for the median of a Gamma distribution. The median must be computed numerically. In this experiment, we approximate the posterior median using 1000 samples from the true posterior Gamma distribution and the torch.median() function.

Setup Our neural architecture employs SetTransformer as the summary network configured to an output dimensionality of 64. A MLP with 2 hidden layers of 32 neurons each serves as our inference network.

During training, we use the Adam optimizer with default settings, set the epoch count to 5000, and the simulation batch size per epoch to 500. The range of observation size [$\mathcal{O}$min, $\mathcal{O}$max] is set between 1 and 20. The chosen range for the power of the loss function, α, denoted as $\mathcal{A}$, spans from 0.25 to 2. Values of α below 0.25 were avoided due to vanishing gradients, which hindered the network's learning.

	$\alpha = 0.25$	$\alpha = 1$	$\alpha = 2$
MSE from mean	1.07e-3±2.84e-5	2.30e-4±1.68e-5	**1.81e-4±9.24e-6**
MSE from median	4.31e-4±2.45e-5	**8.15e-5±1.30e-5**	6.68e-4±1.63e-5
MSE from mode	**1.23e-4±9.88e-6**	7.39e-4±1.41e-5	2.60e-3±2.69e-5

Table 5.5: Mean Squared Error (MSE) and standard deviation over 100 runs for inferring mean, median, and mode statistics at α values of 0.25, 1, and 2. Each cell contains the mean MSE ± one standard deviation, reported in scientific notation.

To evaluate the inference performance, we generate a new simulation dataset of size 1000 and consider a discrete set of 10 values for the parameter α, ranging from 0.25 to 2. For each value of α in this discrete set, we compute our predictions and assess the discrepancy between the true statistics and our predictions using the MSE metric. To ensure the robustness and reliability of our results, we repeat the evaluation process 100 times.

Our findings are presented through a figure that clearly illustrates the error trend for each true statistic as α varies from 0.25 to 2. Additionally, we provide a table that directly shows the numerical mean MSE values and their corresponding standard deviations between our predictions and each true statistic for the considered range of α values. The mean and standard deviation of this score are reported over 100 replications.

Results Analysis Table 5.5 and Figure 5.3 demonstrate results consistent with our expectations. By changing the power of the loss function (α) used during training from 0.25 to 2, we observe that when α is close to zero, our result is nearest to the posterior mode. As α approaches 1, our result is closest to the posterior median, and when α reaches 2, we are closest to the posterior mean. These results clearly demonstrate the effectiveness of our NBPE method in outputting the preferred point estimate at inference time. After training our network with a reasonable range of α values, we can obtain the desired statistic of interest at inference time.

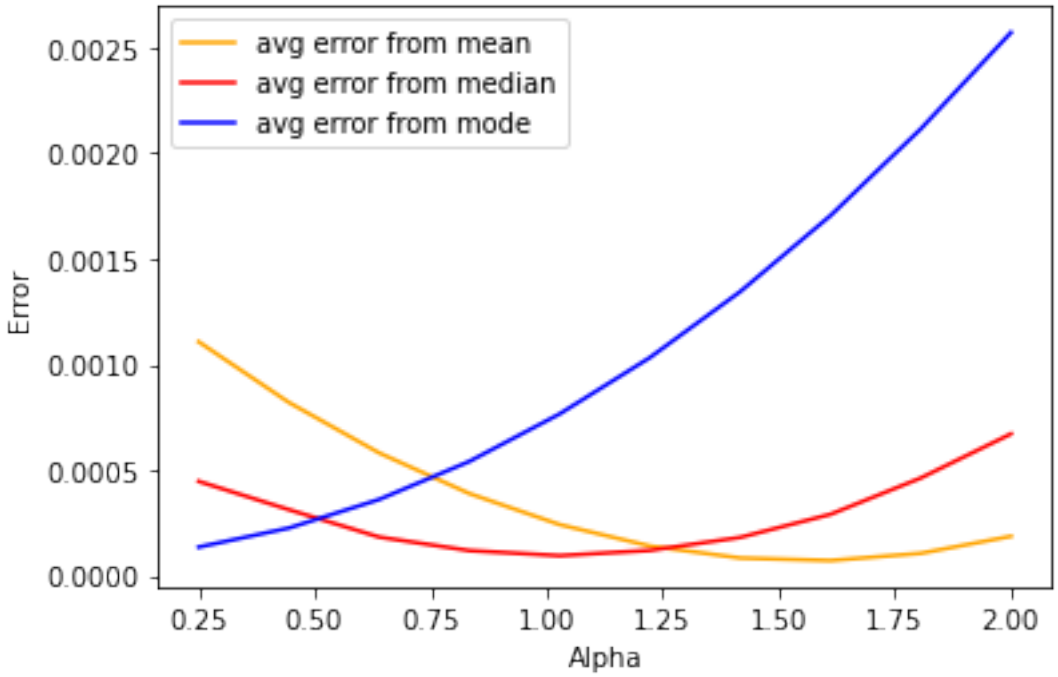


Figure 5.3: Average error between the predicted and true posterior statistics (mean, median, and mode) for the Poisson-Gamma model as a function of the loss power α. The orange line represents the average error from the true posterior mean, the red line represents the average error from the true posterior median, and the blue line represents the average error from the true posterior mode. As α increases from 0.25 to 2.00, the predicted statistic transitions from being closest to the mode (lowest red line error) to the median (lowest orange line error) and finally to the mean (lowest blue line error), demonstrating the ability of the proposed method to target different preferred posterior statistics by varying α.

6. Discussion

In this final chapter, we will first discuss the limitations of our work and future work directions. Then, we will make conclusions about the thesis.

6.1 Limitations & future work

Limited experiments for flexible point estimation

In our work, we tested NBPE's ability to do flexible point estimation on three different models. However, these toy models always have closed-form mean, median, and mode, making the experiments straightforward to analyze. We do not demonstrate this ability on any simulator models, and this is where SBI can show its true value. Thus, one future work must be more experiments on this ability, especially on simulator models.

Neural network is not powerful enough

In our work, NBPE employs a Deepsets or SetTransformer as the summary network and an MLP as the inference network. Our method's choice of the summary network looks good enough, while the inference network looks too simple. We can see clearly from our experiments in the toy Gaussian model that when the neural network has converged, our method cannot exceed NPE. Thus, we worry that our method will likely underfit if we meet some very complex models. In the future, we can explore more advanced network architectures for NBPE, especially the inference network.

The optimization method is simple

In all our experiments, we can see that we can not set the power of loss α to a minimal value (typically < 0.25). This is because, in that case, the network will become extremely hard to train. However, the consequence is that we sometimes cannot clearly observe whether our predicted value will be close to the mode when α approaches 0. Thus, in order to extend the min value we can choose for α, one possible way could be to enrich our optimization method; in our work, we just use the default Adam op-

timizer; in future work, we can employ more training tricks on that such as learning rate scheduling.

6.2 Comparison with BAM

Our work shares some similarities with Bayesian Amortized Decision Making (BAM) [Gorecki et al., 2023], as both methods aim to train an end-to-end neural network to find the best solutions. The main difference between these two works is that we introduce the variable loss function $|\theta_i - \theta_{\text{pred}}|^\alpha$, which enables flexible point estimation by adjusting the value of α. This allows NBPE to target different posterior statistics (e.g., mean, median, mode) using a single pretrained network without additional computations.

6.3 Conclusions

In this thesis, we explore applying the SBI method directly to point estimation. Unlike other SBI methods that first approximate the posterior and then require additional steps to compute point estimations (e.g., using analytical expressions or sampling-based methods), our approach learns an end-to-end model to output the point estimations directly. Furthermore, we introduce a novel training method that enables flexible point estimation, allowing us to target different posterior statistics, such as mean, median, or mode, without extra computational cost during inference time.

More precisely, we propose NBPE (Neural Amortization of Bayesian Point Estimation), a novel neural SBI method for point estimation. By utilizing the powerful features of neural networks, we amortize the inference process and put most of the computational cost on training networks. Besides, utilizing a variable loss function $|\theta_i - \theta_{\text{pred}}|^\alpha$ enables direct prediction of the desired posterior statistic by simply adjusting the value of α. Experiments on toy and simulator models show that NBPE can perform accurate and flexible point estimation.

For future work, several ways could be explored to further improve NBPE. Firstly, more comprehensive experiments focusing on flexible point estimation should be conducted since the current experiments may not be entirely convincing. Secondly, exploring more complex model architectures could potentially enhance NBPE's performance. Thirdly, trying some advanced optimization tricks could allow NBPE to explore a broader range of α values. We believe our work has the potential to be applied in real-world scenarios and contribute to the SBI community.

Bibliography

Justin Alsing, Benjamin Wandelt, and Stephen Feeney. Massive optimal data compression and density estimation for scalable, likelihood-free inference in cosmology. *Monthly Notices of the Royal Astronomical Society*, 477(3):2874–2885, 2018.

Justin Alsing, Thomas DP Edwards, and Benjamin Wandelt. Optimal simulation-based Bayesian decisions. *arXiv preprint arXiv:2311.05742*, 2023.

Andrei Atanov, Arsenii Ashukha, Kirill Struminsky, Dmitry P. Vetrov, and Max Welling. The Deep Weight Prior. In *International Conference on Learning Representations*, 2019.

Divya Banesh, Nishant Panda, Ayan Biswas, Luke Van Roekel, Diane Oyen, Nathan Urban, Michael Grosskopf, Jonathan Wolfe, and Earl Lawrence. Fast Gaussian Process Estimation for Large-Scale in Situ Inference using Convolutional Neural Networks. In *2021 IEEE International Conference on Big Data*, pages 3731–3739. IEEE, 2021.

Mark A Beaumont, Wenyang Zhang, and David J Balding. Approximate Bayesian computation in population genetics. *Genetics*, 162(4):2025–2035, 2002.

Mark A Beaumont, Jean-Marie Cornuet, Jean-Michel Marin, and Christian P Robert. Adaptive approximate Bayesian computation. *Biometrika*, 96(4):983–990, 2009.

Michael Betancourt. Probabilistic Modeling and Statistical Inference. *https://betanalpha.github.io/assets/case_studies/modeling_and_inference.html*, 2019.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational Inference: A Review for Statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

Michael GB Blum, Maria Antonieta Nunes, Dennis Prangle, and Scott A Sisson. A Comparative Review of Dimension Reduction Methods in Approximate Bayesian Computation. *Statistical Science*, 28(2):189–208, 2013.

Fernando V Bonassi and Mike West. Sequential Monte Carlo with Adaptive Weights for Approximate Bayesian Computation. *Bayesian Analysis*, (10):171–187, 2015.

Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. *Handbook of markov chain monte carlo*. CRC press, 2011.

Edgars Butanovs, Aleksejs Zolotarjovs, Alexei Kuzmin, and Boris Polyakov. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment. 2021.

Jeffrey Chan, Valerio Perrone, Jeffrey Spence, Paul Jenkins, Sara Mathieson, and Yun Song. A Likelihood-Free Inference Framework for Population Genetic Data using Exchangeable Neural Networks. In *Advances in Neural Information Processing Systems*, volume 31, pages 8594–8605, 2018.

Tom Charnock, Guilhem Lavaux, and Benjamin D Wandelt. Automatic physical inference with information maximizing neural networks. *Physical Review D*, 97(8): 083004, 2018.

Ki H Chon and Richard J Cohen. Linear and Nonlinear ARMA Model Parameter Estimation Using an Artificial Neural Network. *IEEE Transactions on Biomedical Engineering*, 44(3):168–174, 1997.

Kyle Cranmer, Johann Brehmer, and Gilles Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117(48):30055–30062, 2020.

Michael Creel. Neural Nets for Indirect Inference. *Econometrics and Statistics*, 2: 36–49, 2017.

Giuseppe De Nicolao, Giovanni Sparacino, and Claudio Cobelli. Nonparametric input estimation in physiological systems: Problems, methods, and case studies. *Automatica*, 33(5):851–870, 1997.

Paul Fearnhead and Dennis Prangle. Constructing Summary Statistics for Approximate Bayesian Computation: Semi-automatic ABC. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 74(3):419–474, 2012.

Tomas Geffner, George Papamakarios, and Andriy Mnih. Compositional Score Modeling for Simulation-Based Inference. In *International Conference on Machine Learning*, pages 11098–11116. PMLR, 2023.

Andrew B. Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian Data Analysis*. Chapman and Hall/CRC Press, 3rd edition, 1995.

Charles J Geyer. Practical Markov Chain Monte Carlo. *Statistical Science*, pages 473–483, 1992.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

Mila Gorecki, Jakob H Macke, and Michael Deistler. Amortized Bayesian Decision Making for simulation-based models. *arXiv preprint arXiv:2312.02674*, 2023.

David Greenberg, Marcel Nonnenmacher, and Jakob Macke. Automatic posterior transformation for likelihood-free inference. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2404–2414. PMLR, 2019.

Joeri Hermans. *Advances in Simulation-Based Inference: Towards the automation of the Scientific Method through Learning Algorithms*. PhD thesis, ULiège-University of Liège [Faculty of Applied Sciences], Belgium, 2022.

Joeri Hermans, Volodimir Begy, and Gilles Louppe. Likelihood-free MCMC with Amortized Approximate Ratio Estimators. In *International Conference on Machine Learning*, pages 4239–4248. PMLR, 2020.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural computation*, 9(8):1735–1780, 1997.

George Karabatsos and Fabrizio Leisen. An approximate likelihood perspective on ABC methods. *Statistics Surveys*, 12:66–104, 2018.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, volume 25, 2012.

Simon Lacoste-Julien, Ferenc Huszár, and Zoubin Ghahramani. Approximate inference for the loss-calibrated bayesian. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 416–424. JMLR Workshop and Conference Proceedings, 2011.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep Learning. *nature*, 521(7553): 436–444, 2015.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R Kosiorek, Seungjin Choi, and Yee Whye Teh. Set Transformer: A Framework for Attention-based Permutation-invariant Neural Networks. In *International Conference on Machine Learning*, pages 3744–3753. PMLR, 2018.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising Sequence-to-Sequence Pre-training for Natural language generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, 2020.

Jarno Lintusaari, Michael U Gutmann, Ritabrata Dutta, Samuel Kaski, and Jukka Corander. Fundamentals and Recent Developments in Approximate Bayesian Computation. *Systematic biology*, 66(1):e66–e82, 2017.

Jan-Matthis Lückmann. *Simulation-Based Inference for Neuroscience and Beyond*. PhD thesis, Universität Tübingen, 2022.

Jan-Matthis Lueckmann, Pedro J Goncalves, Giacomo Bassetto, Kaan Öcal, Marcel Nonnenmacher, and Jakob H Macke. Flexible statistical inference for mechanistic models of neural dynamics. In *Advances in Neural Information Processing Systems*, volume 30, pages 1289–1299, 2017.

Jan-Matthis Lueckmann, Giacomo Bassetto, Theofanis Karaletsos, and Jakob H Macke. Likelihood-free inference with emulator networks. In *Proceedings of The 1st Symposium on Advances in Approximate Bayesian Inference*, volume 96, pages 32–53. PMLR, 2019.

Jan-Matthis Lueckmann, Jan Boelts, David Greenberg, Pedro Goncalves, and Jakob Macke. Benchmarking Simulation-Based Inference. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130, pages 343–351. PMLR, 2021.

Paul Marjoram, John Molitor, Vincent Plagnol, and Simon Tavaré. Markov chain Monte Carlo without likelihoods. *Proceedings of the National Academy of Sciences of the United States of America*, 100(26):15324–15328, 2003.

Iain Murray. *Advances in Markov chain Monte Carlo methods*. PhD thesis, Gatsby computational neuroscience unit, University College London, 2007.

Lorenzo Pacchiardi and Ritabrata Dutta. Likelihood-Free Inference with Generative Neural Networks via Scoring Rule Minimization. *arXiv preprint arXiv:2205.15784*, 2022.

George Papamakarios. *Neural density estimation and likelihood-free inference*. PhD thesis, School of Informatics, University of Edinburgh, 2019.

George Papamakarios and Iain Murray. Fast ε-free Inference of Simulation Models with Bayesian Conditional Density Estimation. In *Advances in Neural Information Processing Systems*, volume 29, pages 1028–1036, 2016.

George Papamakarios, Iain Murray, and Theo Pavlakou. Masked Autoregressive Flow for Density Estimation. In *Advances in Neural Information Processing Systems*, volume 30, pages 2335–2344, 2017.

Dennis Prangle. Adapting the ABC distance function. *Bayesian Analysis*, 12:289–309, 2017.

Stefan T Radev, Ulf K Mertens, Andreas Voss, Lynton Ardizzone, and Ullrich Köthe. BayesFlow: Learning complex stochastic models with invertible neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 33(4):1452–1466, 2020.

Stefan T Radev, Marvin Schmitt, Lukas Schumacher, Lasse Elsemüller, Valentin Pratz, Yannik Schälte, Ullrich Köthe, and Paul-Christian Bürkner. BayesFlow: Amortized Bayesian Workflows With Neural Networks. *Journal of Open Source Software*, 2023.

Poornima Ramesh, Jan-Matthis Lueckmann, Jan Boelts, Álvaro Tejero-Cantero, David S Greenberg, Pedro J Gonçalves, and Jakob H Macke. GATSBI: Generative Adversarial Training for Simulation-Based Inference. In *International Conference on Learning Representations*, 2022.

Oliver Ratmann, Ole Jørgensen, Trevor Hinkley, Michael Stumpf, Sylvia Richardson, and Carsten Wiuf. Using Likelihood-Free Inference to Compare Evolutionary Dynamics of the Protein Networks of H. pylori and P. falciparum. *PLoS Computational Biology*, 3(11):e230, 2007.

Johann Rudi, Julie Bessac, and Amanda Lenzi. Parameter Estimation with Dense and Convolutional Neural Networks Applied to the FitzHughâNagumo ODE. In *Proceedings of the 2nd Mathematical and Scientific Machine Learning Conference*, volume 145, pages 781–808. PMLR, 2022.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Scott A Sisson, Yanan Fan, and Mark M Tanaka. Sequential Monte Carlo without likelihoods. *Proceedings of the National Academy of Sciences*, 104(6):1760–1765, 2007.

Scott A Sisson, Yanan Fan, and Mark Beaumont. *Handbook of Approximate Bayesian Computation*. Chapman and Hall/CRC, 2018.

Alvaro Tejero-Cantero, Jan Boelts, Michael Deistler, Jan-Matthis Lueckmann, Conor Durkan, Pedro J Gonçalves, David S Greenberg, and Jakob H Macke. SBI–A toolkit for simulation-based inference. *Journal of Open Source Software*, 5(52):2505, 2020.

Michael E Tipping. Bayesian inference: An introduction to principles and practice in machine learning. In *Advanced Lectures on Machine Learning*, pages 41–62. 2004.

Jakub Tomczak and Max Welling. VAE with a VampPrior. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics*, volume 84, pages 1214–1223. PMLR, 2018.

Malavika Vasist, François Rozet, Olivier Absil, Paul Mollière, Evert Nasedkin, and Gilles Louppe. Neural posterior estimation for exoplanetary atmospheric retrieval. *Astronomy & Astrophysics*, 672:A147, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. volume 30, 2017.

Christina Winkler, Daniel Worrall, Emiel Hoogeboom, and Max Welling. Learning Likelihoods with Conditional Normalizing Flows. *arXiv preprint arXiv:1912.00042*, 2019.

Samuel Wiqvist, Jes Frellsen, and Umberto Picchini. Sequential Neural Posterior and Likelihood Approximation. *arXiv preprint arXiv:2102.06522*, 2021.

Simon N Wood. Statistical inference for noisy nonlinear ecological dynamic systems. *Nature*, 466(7310):1102–1104, 2010.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep Sets. In *Advances in Neural Information Processing Systems*, volume 30, pages 3391–3401, 2017.

Cheng Zhang, Judith Bütepage, Hedvig Kjellström, and Stephan Mandt. Advances in Variational Inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):2008–2026, 2018.

Appendix A.

A.1 Code and hardware settings for experiments

All code related to the proposed methodology and experiments in this thesis is available in a public GitHub repository at https://github.com/swagylee/amortized-estimation. This repository contains the necessary scripts, data, and documentation to reproduce the results presented in this work. All experiments were conducted on a MacBook Pro 13-inch, 2019 model with the following specifications:

- CPU: 2.3 GHz 4-cores Intel Core i5
- GPU: Intel Iris Plus Graphics 655 with 1536 MB of memory
- RAM: 8 GB 2133 MHz LPDDR3

A.2 The derivation of the Evidence Lower Bound

Starting from the KL divergence minimization objective:

$$\begin{aligned} \lambda^* &= \arg\min_{\lambda} \ \mathrm{KL}(q(\theta|\mathbf{x},\lambda)|p(\theta|\mathbf{x},\phi)) \\ &= \arg\min_{\lambda} \ \int_\theta q(\theta|\mathbf{x},\lambda)\log\frac{q(\theta|\mathbf{x},\lambda)}{p(\theta|\mathbf{x},\phi)}d\theta \end{aligned} \tag{A.1}$$

We can expand the fraction inside the logarithm using Bayes' theorem:

$$p(\theta|\mathbf{x},\phi) = \frac{p(\mathbf{x}|\theta,\phi)p(\theta)}{p(\mathbf{x}|\phi)} \ \Rightarrow \frac{q(\theta|\mathbf{x},\lambda)}{p(\theta|\mathbf{x},\phi)} \ = \frac{q(\theta|\mathbf{x},\lambda)p(\mathbf{x}|\phi)}{p(\mathbf{x}|\theta,\phi)p(\theta)} \tag{A.2}$$

Substituting this back into the KL divergence expression:

$$\begin{aligned} \mathrm{KL}(q(\theta|\mathbf{x},\lambda)|p(\theta|\mathbf{x},\phi)) &= \int_\theta q(\theta|\mathbf{x},\lambda)\log\frac{q(\theta|\mathbf{x},\lambda)p(\mathbf{x}|\phi)}{p(\mathbf{x}|\theta,\phi)p(\theta)}d\theta \\ &= \int_\theta q(\theta|\mathbf{x},\lambda)\log\frac{q(\theta|\mathbf{x},\lambda)}{p(\mathbf{x}|\theta,\phi)p(\theta)}d\theta + \int_\theta q(\theta|\mathbf{x},\lambda)\log p(\mathbf{x}|\phi)d\theta \\ &= \int_\theta q(\theta|\mathbf{x},\lambda)\log\frac{q(\theta|\mathbf{x},\lambda)}{p(\mathbf{x}|\theta,\phi)p(\theta)}d\theta + \log p(\mathbf{x}|\phi)\int_\theta q(\theta|\mathbf{x},\lambda)d\theta \end{aligned} \tag{A.3}$$

Since $q(\theta|\mathbf{x},\lambda)$ is a probability distribution, its integral over the entire space is 1. Therefore:

$$\mathrm{KL}(q(\theta|\mathbf{x},\lambda)|p(\theta|\mathbf{x},\phi)) = \int_\theta q(\theta|\mathbf{x},\lambda)\log\frac{q(\theta|\mathbf{x},\lambda)}{p(\mathbf{x}|\theta,\phi)p(\theta)}d\theta + \log p(\mathbf{x}|\phi) \tag{A.4}$$

Rearranging the terms:

$$\log p(\mathbf{x}|\phi) = \mathrm{KL}(q(\theta|\mathbf{x},\lambda)|p(\theta|\mathbf{x},\phi)) - \int_\theta q(\theta|\mathbf{x},\lambda)\log\frac{q(\theta|\mathbf{x},\lambda)}{p(\mathbf{x}|\theta,\phi)p(\theta)}d\theta \tag{A.5}$$

The second term on the right-hand side is the negative of the ELBO:

$$\begin{aligned}\mathrm{ELBO}(q) &= \int_\theta q(\theta|\mathbf{x},\lambda)\log\frac{p(\mathbf{x}|\theta,\phi)p(\theta)}{q(\theta|\mathbf{x},\lambda)}d\theta \\ &= \int_\theta q(\theta|\mathbf{x},\lambda)\log p(\mathbf{x}|\theta,\phi)d\theta - \int_\theta q(\theta|\mathbf{x},\lambda)\log\frac{q(\theta|\mathbf{x},\lambda)}{p(\theta)}d\theta \\ &= \mathbb{E}_{\theta\sim q(\theta|\mathbf{x},\lambda)}[\log p(\mathbf{x}|\theta,\phi)] - \mathbb{E}_{\theta\sim q(\theta|\mathbf{x},\lambda)}\left[\log\frac{q(\theta|\mathbf{x},\lambda)}{p(\theta)}\right]\end{aligned} \tag{A.6}$$

Therefore, the log-evidence can be expressed as:

$$\log p(\mathbf{x}|\phi) = \mathrm{KL}(q(\theta|\mathbf{x},\lambda)|p(\theta|\mathbf{x},\phi)) + \mathrm{ELBO}(q) \tag{A.7}$$

Since the KL divergence is always non-negative, the ELBO serves as a lower bound on the log-evidence. Maximizing the ELBO with respect to the variational parameters λ is equivalent to minimizing the KL divergence between the approximate posterior $q(\theta|\mathbf{x},\lambda)$ and the true posterior $p(\theta|\mathbf{x},\phi)$.

A.3 Proof for proposition 1

To prove the proposition that the choice of α in the loss function targets the mean, median, and mode of the posterior distribution, respectively, for $\alpha = 2, \alpha = 1$, and $\alpha \to 0$.

For $\alpha = 2$(Targeting the Mean), the loss function is the Mean Squared Error (MSE):

$$L(\phi,\psi) = \frac{1}{N}\sum_{i=1}^{N}\left(\theta_i - \mathcal{I}_\psi\left(S\left(\mathbf{x}_i\right),2\right)\right)^2. \tag{A.8}$$

To find the value of $\mathcal{I}\psi$ that minimizes $L(\phi,\psi)$, we take the derivative w.r.t $\mathcal{I}\psi$ and set it to zero:

$$\frac{\partial L}{\partial \mathcal{I}_\psi} = -\frac{2}{N}\sum_{i=1}^{N}\left(\theta_i - \mathcal{I}_\psi\left(S\left(\mathbf{x}_i\right),2\right)\right) = 0 \Longrightarrow \mathcal{I}_\psi = \frac{1}{N}\sum_{i=1}^{N}\theta_i \tag{A.9}$$

Therefore, minimizing the MSE loss ($\alpha = 2$) yields the mean $\frac{1}{N}\sum_{i=1}^{N}\theta_i$ of the true parameters.

For $\alpha = 1$(Targeting the Median), the loss function is the Mean Absolute Error (MAE):

$$L(\phi,\psi) = \frac{1}{N}\sum_{i=1}^{N}\left|\theta_i - \mathcal{I}_\psi\left(S\left(\mathbf{x}_i\right),1\right)\right| \tag{A.10}$$

Let's consider a set of N observations $\theta_1, \theta_2, \ldots, \theta_N$ sorted in ascending order. The median m is defined as: $m = \begin{cases} \theta_{\frac{N+1}{2}}, & \text{if } N \text{ is odd} \\ \frac{\theta_{\frac{N}{2}} + \theta_{\frac{N}{2}+1}}{2}, & \text{if } N \text{ is even} \end{cases}$

Now, let's consider an arbitrary point $a \neq m$. We will show that moving a towards m will decrease the sum of absolute deviations.

Case 1: $a < m$ If we increase a by a small amount $\epsilon > 0$ such that $a + \epsilon \leq m$, the sum of absolute deviations will change by:

$$\sum_{i=1}^{N} |\theta_i - (a+\epsilon)| - \sum_{i=1}^{N} |\theta_i - a| = \sum_{\theta_i < a} \epsilon + \sum_{a \leq \theta_i < a+\epsilon} (\theta_i - a - \epsilon) + \sum_{a+\epsilon \leq \theta_i} (-\epsilon)$$

Since $a < m$, there are more points greater than or equal to a than there are points less than a. Therefore, the net change in the sum of absolute deviations will be negative, indicating a decrease.

Case 2: $a > m$ Similarly, if we decrease a by a small amount $\epsilon > 0$ such that $a - \epsilon \geq m$, the sum of absolute deviations will change by:

$$\sum_{i=1}^{N} |\theta_i - (a-\epsilon)| - \sum_{i=1}^{N} |\theta_i - a| = \sum_{\theta_i < a-\epsilon} (-\epsilon) + \sum_{a-\epsilon \leq \theta_i < a} (a - \epsilon - \theta_i) + \sum_{a \leq \theta_i} \epsilon$$

Since $a > m$, there are more points less than or equal to a than there are points greater than a. Therefore, the net change in the sum of absolute deviations will again be negative, indicating a decrease.

In both cases, moving a towards m decreases the sum of absolute deviations. This process can be repeated until a reaches m, at which point no further improvement can be made. Therefore, the median m minimizes the MAE.

For $\alpha \to 0$(Targeting the mode), the loss function converges to the indicator function:

$$\lim_{\alpha \to 0} L(\phi, \psi) = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1} \left(\theta_i \neq \mathcal{I}_\psi \left(S \left(\mathbf{x}_i \right), 0 \right) \right)$$

where $\mathbf{1}(\cdot)$ is the indicator function that equals 1 if the condition inside the parentheses is true and 0 otherwise.

To minimize this loss function, we need to maximize the number of exact matches between the true parameters θ_i and the predictions $\mathcal{I}_\psi \left(S \left(\mathbf{x}_i \right), 0 \right)$. In other words, we want to find the value that appears most frequently in the set of true parameters $\theta_1, \theta_2, \ldots, \theta_N$.

In summary, by varying the value of α in the loss function $L(\phi, \psi) = \frac{1}{N} \sum_{i=1}^{N} |\theta_i - \mathcal{I}_\psi \left(S \left(\mathbf{x}_i \right), \alpha \right)|^\alpha$, we can target different statistical measures of the posterior distribution $p(\theta|x)$. Setting $\alpha = 2$ targets the mean, $\alpha = 1$ targets the median, and $\alpha \to 0$ targets the mode of the posterior distribution.

A.4 Additional experiments for flexible point estimation

In this section, we present further experiments to investigate the NBPE's ability of outputting flexible point estimations in two different models: the Beta-Bernoulli model and the Gamma-Exponential model.

Beta-Bernoulli model In this model, a binary random variable is also known as a Bernoulli variable and follows the Bernoulli distribution. The parameter of the Bernoulli distribution is $\pi \in [0, 1]$, the probability of the event happening in each trial. A common goal is to learn the parameter π given a sequence of observations $\left\{\mathbf{x}_o^1, \ldots, \mathbf{x}_o^N\right\}$. Here, $\mathbf{x}_o^i = 1$ means that the event happens (a 'hit' or a 'success') and $\mathbf{x}_o^i = 0$ means that the event does not happen (a 'miss' or a 'fail') on the i-th trial. The likelihood is

$$p\left(\mathbf{x}_o^i = 1\right) = \pi$$
$$p\left(\mathbf{x}_o^i = 0\right) = 1 - \pi.$$

and we select a prior $p(\pi)$ over the Bernoulli parameter (i.e., the probability of the event).

If the prior is chosen to be a Beta prior, $\text{Beta}\left(\alpha_0, \beta_0\right)$, with $\alpha_0, \beta_0 > 0$, then the posterior is also a Beta distribution,

$$p\left(\pi|\mathbf{x}_{\text{obs}}\right) = \text{Beta}\left(\pi|\alpha_{\text{post}}\,, \beta_{\text{post}}\,\right), \quad \alpha_{\text{post}}\, = \alpha_0 + n_{\text{hit}}\,, \beta_{\text{post}}\, = \beta_0 + n_{\text{miss}}$$

where $n_{\text{hit}} \equiv \sum_{i=1}^{N} \mathbf{x}_o^i$ is the number of 'hits' and $n_{\text{miss}}\, \equiv \sum_{i=1}^{N} \left(1 - \mathbf{x}_o^i\right)$ is the number of 'misses'.

When we have a Bernoulli likelihood and a Beta prior, the posterior distribution is a Beta distribution. Specifically, the posterior distribution is given by:

$$\text{Beta}(\alpha_{\text{post}}, \beta_{\text{post}}) = \text{Beta}(\alpha_0 + n_{\text{hit}}, \beta_0 + n_{\text{miss}})$$

where n_{hit} is the number of successes and n_{miss} is the number of failures.

The mode, mean, and median of the Beta distribution are given by: Mode: $\frac{\alpha_{\text{post}}-1}{\alpha_{\text{post}}+\beta_{\text{post}}-2}$ (for $\alpha_{\text{post}}, \beta_{\text{post}} > 1$) Mean: $\frac{\alpha_{\text{post}}}{\alpha_{\text{post}}+\beta_{\text{post}}}$ Median: Approximated by $\frac{\alpha_{\text{post}}-\frac{1}{3}}{\alpha_{\text{post}}+\beta_{\text{post}}-\frac{2}{3}}$ (for $\alpha_{\text{post}}, \beta_{\text{post}} > 1$)

Gamma-Exponential model The Gamma-Exponential model is commonly used to model waiting times or inter-arrival times in various applications. In this model, the waiting times are assumed to follow an Exponential distribution, while the rate

parameter of the Exponential distribution is modeled using a Gamma distribution as the prior.

The likelihood of the model is given by the Exponential distribution. The probability density function of the Exponential distribution is:

$$f(\mathbf{x}|\lambda) = \lambda e^{-\lambda \mathbf{x}}$$

where $\mathbf{x}$ is the waiting time and λ is the rate parameter.

The prior distribution for the rate parameter λ is assumed to be a Gamma distribution with shape parameter α and rate parameter β:

$$g(\lambda|\alpha,\beta) = \frac{\beta^\alpha}{\Gamma(\alpha)}\lambda^{\alpha-1}e^{-\beta\lambda}$$

Given observed waiting times $\mathbf{x}_o^1, \ldots, \mathbf{x}_o^N$, the posterior distribution of λ is also a Gamma distribution with updated parameters: 1.Shape parameter of the posterior $(\alpha_{\text{post}}) = \alpha + n$, where n is the number of observations. 2.Rate parameter of the posterior $(\beta_{\text{post}}) = \beta + \sum_{i=1}^{N} \mathbf{x}_o^i$.

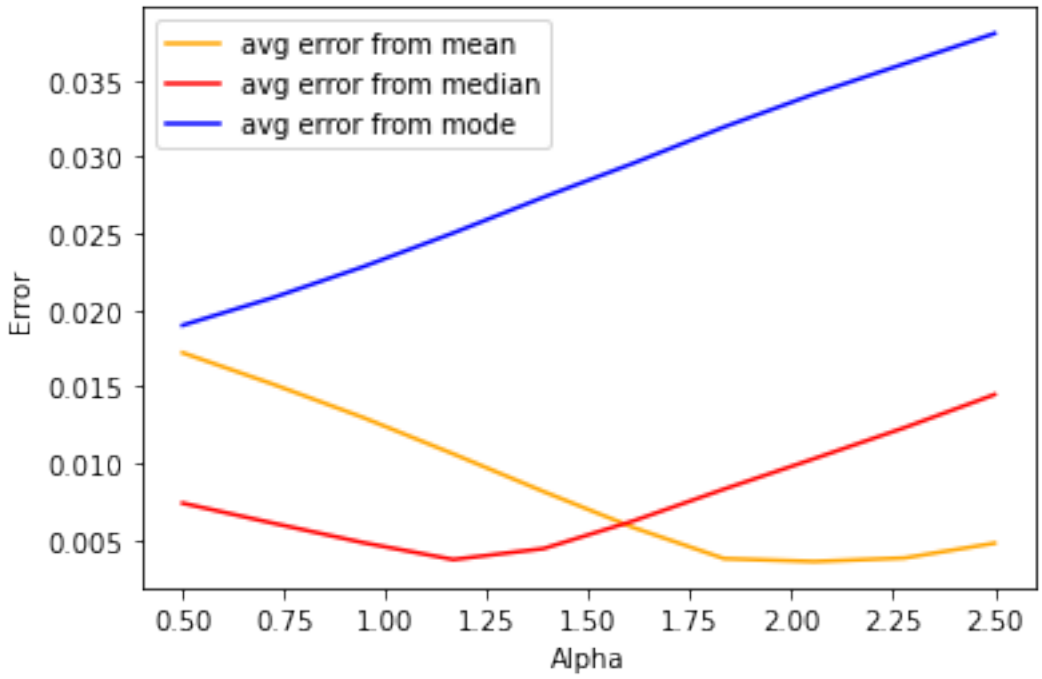
The posterior mean of λ can be calculated as: $\frac{\alpha_{\text{post}}}{\beta_{\text{post}}}$. The posterior median of λ has no closed form. The posterior mode of λ can be calculated as: $\frac{\alpha_{\text{post}}-1}{\beta_{\text{post}}}$ (for $\alpha_{\text{post}} > 1$).

Experiments set up For Beta-Bernoulli model, we use Deepsets as summary network with $D_{\text{summary}} = 8$, a two hidden layers MLP as inference network and train NBPE for 5000 epochs with default Adam optimizer. And for simulation parameters we set simulation size per epoch to 96 and observation size from 10 to 20.And the range of α is from 0.5 to 2.

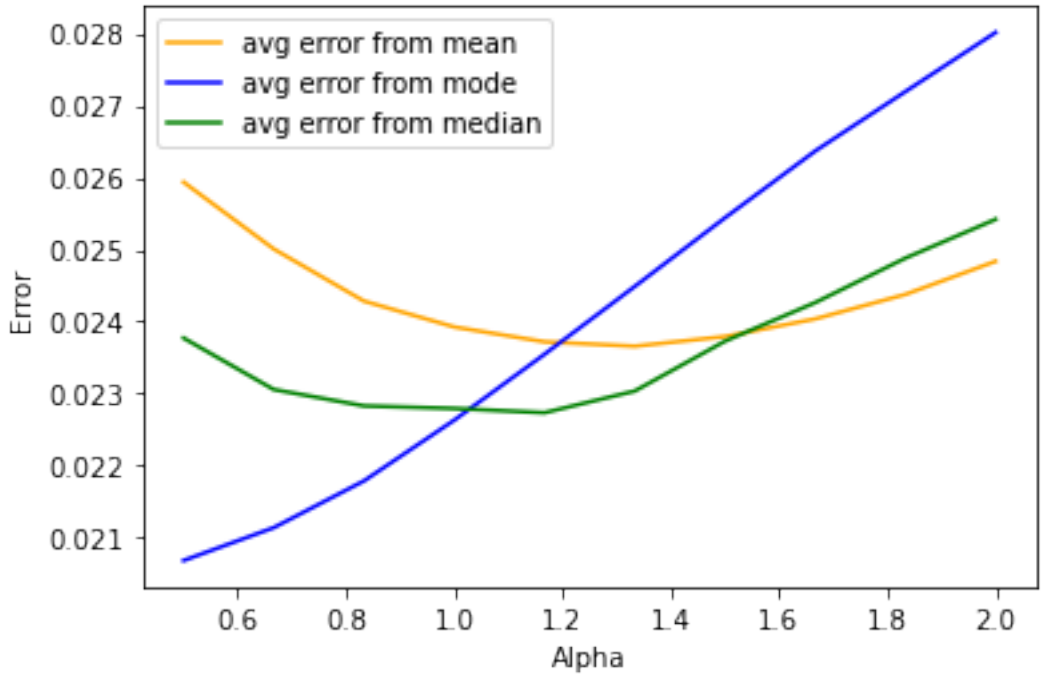
For Gamma-Exponential model, we use Settransformer as summary network this time with $D_{\text{summary}} = 16$, a two hidden layers MLP as inference network and train NBPE for 2000 epochs with default Adam optimizer. And for simulation parameters we set simulation size per epoch to 96 and observation size from 20 to 30. And the range of α is also from 0.5 to 2.

Results analysis Figure A.1 presents the results of the Beta-Bernoulli and Gamma-Exponential models. The result on Gamma-Exponential model closely aligns with our expectations regarding the output change as the parameter α varies. The result on Beta-Bernoulli model also shows a similar trend, although there is some uncertainty about the point estimation's proximity to the mode when $\alpha \to 0$. Nevertheless, the overall trend in both models is promising.

These additional experiments further validate our method's ability to perform preferred point estimations effectively. The results demonstrate the robustness and applicability of our approach across different models and scenarios.



(a)



(b)

Figure A.1: Comparison of the average error from the mean, median, and mode for different values of α in the (a) Beta-Bernoulli model and (b) Gamma-Exponential model.