Simulate or Emulate? Empirical Evaluation of Deep Generative Modelling Approaches for Bayesian Inference.

Herman Franclin Tesso (herman@aims.ac.za)
African Institute for Mathematical Sciences (AIMS)

Supervised by: Dr Elizaveta Semenova
Imperial College London, UK
Co-supervised by: Dr Ayush Bharti
Aalto University, Finland
Co-supervised by: Prof Seth Flaxman
University of Oxford, UK

24 October 2024

Submitted in partial fulfillment of a structured masters degree at AIMS South Africa

AIMS | African Institute for Mathematical Sciences SOUTH AFRICA

Abstract

MCMC methods are the leading approach for Bayesian inference in epidemiology, especially in the context of spatial models and mechanistic disease transmission models. Although MCMC is asymptotically accurate, it is a case-by-case method that requires costly recomputation for newly observed data and scales inefficiently, particularly in spatial models that use Gaussian Processes (GPs) to represent spatial correlations. To address these limitations, we propose leveraging deep generative models for inference, including a probabilistic diffusion model trained with transformer architectures (Simformer) and a conditional invertible network (cINN) in the form of normalizing flows for posterior estimation within the simulation-based inference framework. These methods circumvent the computational bottlenecks of MCMC, generalize effectively over spatially correlated structures, manage function-valued parameters (Simformer), and provide a faster, scalable solution for spatial inference while amortizing posterior distributions. We demonstrate their performance and flexibility through experiments on spatial grids and ODE-based compartmental models, showcasing their potential as practical tools for approximate inference in real-world interpolation tasks (e.g., spatial modelling) and extrapolation tasks.

Keywords: Bayesian inference, deep generative modelling, simulation-based inference, epidemiology, spatial modelling, compartmental models, Gaussian process prior.

Declaration

I, the undersigned, hereby declare that the work contained in this research project is my original work, and that any work done by others or by myself previously has been acknowledged and referenced accordingly.

[illegible]

Herman Franclin Tesso, 24 October 2024

Contents

1. Introduction

Epidemiology plays a crucial role in understanding the impact of diseases on populations and the strategies for their control. It reveals patterns in the spread of illnesses, providing public health officials with the essential insights needed for effective responses. By analyzing outbreaks—whether seasonal flu or global pandemics—epidemiologists can identify who is most at risk, how diseases propagate through different communities, and what preventive measures can be implemented to avert escalation. Mathematical models are key tools in this field, allowing researchers to simulate and predict disease spread under various scenarios. Two common model types are compartmental models (Brauer, 2008), which categorize populations based on disease status, simplifying complex real-world dynamics into manageable mathematical equations that describe the flow of individuals between compartments, and spatial models (Martínez et al., 2022), which integrate geographic or social factors to study how diseases disseminate across different regions or communities.

For example, consider the 2D grid cells depicted in **Fig.** 1.1, where some cells display observed values (colored cells), while others remain unknown (masked). In practical situations, these cells may represent specific geographical coordinates, such as latitude and longitude, predefined geographic units, existing administrative areas, or pixels of a raster image. The observations can represent various data types: counts aggregated over an area (e.g., the number of disease cases), continuous bounded data (e.g., disease prevalence ranging from 0 to 1), or continuous unbounded data (e.g., rainfall intensity). The latter data types are referred to as spatial data (Anselin et al., 2009), providing insights into phenomena that vary across geographic space. Accurately modelling these data can help bridge gaps where information is lacking, identify critical trends, and enhance predictions that inform policy decisions, resource allocation, and risk management in fields such as epidemiology, environmental science, and public health.

Additionally, consider the compartmental model illustrated in **Fig.** 1.1 (right), which represents population states during a potential epidemic . This dynamic is typically described by a mechanistic model governed by interpretable parameters that may be either constant-in-time or time-varying.

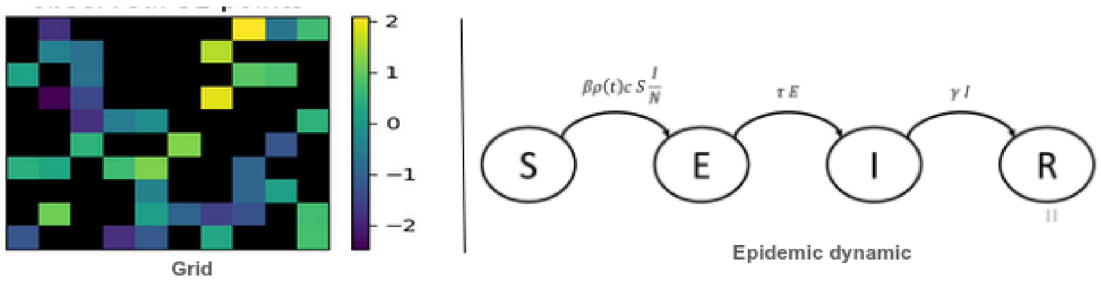

Figure 1.1: Overview of common inference scenarios in epidemiology.

Inferring these parameters from data is crucial, as accurate parameterization allows the model to reflect the dynamics of disease spread more realistically as well as inferring over spatial structures for disease mapping is essential for targeted interventions and resource allocation, ensuring that public health efforts are effectively directed to the areas most at risk. Bayesian inference (Box and Tiao, 2011) is the preferred framework for this purpose, as it systematically incorporates prior knowledge with observed data and allows for uncertainty quantification, which is particularly beneficial during early outbreaks when data may be limited. Compared to frequentist methods, Bayesian approaches provide a more flexible framework for updating beliefs as new data becomes available, making them valuable for navigating the complexities of epidemiological data and enhancing predictive accuracy.

The theoretical asymptotic convergence guarantees of Markov Chain Monte Carlo (MCMC) methods make them among the most reliable options for Bayesian inference, alongside other approximate methods such as variational inference (Blei et al., 2017), Laplace approximation (Shun and McCullagh, 1995), and Expectation propagation (Minka, 2013). Consequently, these methods are often favored for epidemiological inference. Recently, a group of two-stage methods leveraging deep generative models has been proposed to assist MCMC inference involving Gaussian process (GP) priors in the context of small area estimation. For example, PriorVAE (Semenova et al., 2022) learns uncorrelated representations of spatial GP priors for a predefined spatial setting and uses the trained decoder as a surrogate for the original GP during inference. Meanwhile, PriorCVAE (Semenova et al., 2023) allows for the joint encoding of hyperparameters with GP realizations, facilitating their subsequent estimation during inference.

Despite these advancements enabling more efficient inference, reliance on MCMC methods presents certain drawbacks. First, MCMC is a case-based approach, requiring computations to be restarted from scratch for every new observed data. Second, it necessitates the specification of a fixed approximation task. In practice, particularly in spatial modelling, one might want to interactively explore different conditioned distributions on observed outcomes from multiple subsets of sparse locations. Third, MCMC does not scale well with data dimensionality and often struggles with the high autocorrelation present in spatial data, which limits its effectiveness in large-scale spatial applications. Finally, MCMC is slow to converge, especially in high-dimensional parameter spaces.

In this study, we seek solutions to these limitations of MCMC methods by exploring simulation-based inference (SBI) techniques (Papamakarios et al., 2019), a family of approximate inference methods that are well-suited for models with implicit likelihood. We propose using two implementations of deep generative modelling for posterior inference within the SBI framework. These implementations aim to address the challenges of MCMC inference in spatial settings and stochastic models in epidemiology.

The first approach combines the expressive power of diffusion models with the transformer's ability to handle sequence data (**Sec.** 4.1). This model effectively addresses inference tasks involving both parametric and non-parametric likelihood parameters (i.e function-value parameters) and can manage missing data, making it particularly promising for spatial and spatio-temporal inference tasks. Additionally, it produces a single network that can be queried to sample arbitrary conditionals of the joint distribution. This contrasts with MCMC methods, which cannot leverage training experience and require substantial computational effort for each query.The second approach employs flow-based models (**Sec.** 4.2) implemented through specialized invertible networks for conditional density estimation. Originally designed for rapid and amortized inference in ODE-based time series models, we also propose a variant specifically tailored for spatial inference tasks involving Gaussian Processes (GPs) (Rasmussen and Nickisch, 2010). This method yields posterior distributions that are fully compatible with Bayesian interpretation and can be used to assess the uncertainty associated with any estimated quantity.

The primary objective of this work is to evaluate the suitability of these methods for inference in spatial models in epidemiology that rely on GPs, as well as for inference in ODE-based disease models.

Our contribution can be summarise as follow:

- We present two generative modelling approaches as alternatives to MCMC-based methods for Bayesian inference in epidemiology: a diffusion model on top of a transformer architecture (Simformer) and a conditional invertible network (cINN) architecture.
- We demonstrate for the first time the applicability of these methods to epidemiological spatial models with latent Gaussian process priors. In particular, we show that the transformer-based diffusion model can generalize the inference task over the underlying correlation structure to new sets of observed data at any specified locations.
- We showcase the effectiveness of both approaches in inferring constant-in-time and time-varying parameters in ODE-based disease models within a simulation-based inference framework and demonstrate that they amortize the posterior distribution.

Our exploration of SBI methods aims to offer a new perspective on inference tasks in epidemiology and their applications for the foreseeable future. We begin with a mathematical description of the problems in **Section** 2 and outline the foundational components of our approaches in **Section** 3. **Section** 4 provides a detailed workflow description of the methods. We then conduct a series of experiments on various synthetic data in **Section** 5 to demonstrate the applicability and performance of our approaches, focusing on both spatial and ODE-based disease models. Finally, in **Section** 6, we discuss our findings, highlight the limitations, and offer insights into potential directions for future improvement.

2. Problem setting and approach

In this section, we formally define the problems we intend to solve. By providing a clear mathematical formulation, we lay the groundwork for the approaches and methods discussed in later sections. This precise definition of the problems aids in guiding the strategies we use to achieve our objectives.

2.1 Spatial inference for disease mapping

Suppose we have outcome data $y_1, \ldots, y_n$ associated with a set of observed disjoint areas $\{A_i\}$, that together cover the domain of interest $G = \bigcup_{i=1}^{n} A_i$. In applied fields, it is common to employ generalized linear models to provide a unified modelling approach for all such types of outcomes. When considering a mixed-effects model structure, their Bayesian hierarchical formulation can be expressed as follows:

$$\theta \sim p(\theta) \tag{2.1.1}$$
$$f \mid \theta \sim GP\left(\mu, \Sigma(\theta)\right) \tag{2.1.2}$$
$$\eta = X\beta + f \tag{2.1.3}$$
$$y \mid \eta \sim p\left(u^{-1}(\eta), \theta\right) \tag{2.1.4}$$

- **(2.1.1)** outlines the model's hyperparameters, denoted as θ.
- In **(2.1.2)**, f represents the latent Gaussian field that captures the spatial structure. It is common practice to assign Gaussian Process (GP) priors to f, defined by a mean μ and a covariance $\Sigma(\theta)$, which depend on the spatial setting of the problem and the model selected for the random effect.
- In **(2.1.3)**, X denotes the fixed effects design matrix (a set of covariates), β represents the fixed effects, and η is the linear predictor that combines both fixed and random effects.
- Equation **(2.1.4)** presents an observational model, where u is a link function that characterizes the mean of the distribution (e.g., an exponential function for positive data).

A common modelling approach modeling assumption is that the observations y_i are conditionally independent $p\left(y \mid f, \theta\right) = \prod_{i=1}^{n} p\left(y_i \mid \eta(f_i), \theta\right)$. Once the model for f_{GP} has been defined, the linear predictor can be computed as

$$u\left(\mathbb{E}[y \mid f_{GP}]\right) = X\beta + f_{GP}$$

and then linked to the observed data through the likelihood. Since f_{GP} always enters the model via the linear predictor, we can, without loss of generality, consider f_{GP} to have a zero mean and be distributed as $f \sim GP\left(0, \Sigma\right))$. The random effect f_{GP} presents the computational challenge during Bayesian inference. Further, we describe a Transformer-based diffusion approach (**Sec.** 4.1) within the framework of simulation-based inference that effectively manages this computational burden while enhancing inference efficiency and leveraging the ability to generalize to new data. The architecture is further used for natural simulation-based inference tasks involving function-valued parameters (**Sec.** 5.4).

2.2 Inference for ODE-based disease transmission models

We consider a mechanistic model (2.2.1) with parameters θ potentially non-parametric (function-valued parameters) (2.2.2) which stochastically generates samples y from its implicit likelihood $p(y|\theta)$ (ODE).

$$\frac{dS}{dt} = -\beta\frac{SI}{N} \quad , \quad \frac{dI}{dt} = \beta\frac{SI}{N} - \gamma I \quad , \quad \frac{dR}{dt} = \gamma I \tag{2.2.1}$$

$$\frac{dS}{dt} = -\beta(t)\frac{SI}{N} \quad , \quad \frac{dI}{dt} = \beta(t)\frac{SI}{N} - \gamma I \quad , \quad \frac{dR}{dt} = \gamma I \tag{2.2.2}$$

Here $\theta \subset \{\beta, \gamma, \beta(t)\}$ are interpretable parameters described further in **Sec.** 5.3.
After having observed data $y_{1:T}$ (time series), we aim to infer the posterior distribution $p(\theta|y_{1:T})$ of parameters given data. Well-suited to be efficiently addressed by a simulation-based inference approach, we therefore introduce a specialized flow-based model architecture **Sec.** 4.2 trained on synthetic simulation (θ_n, y_n) , primarily designed for amortized inference on compartmental models in epidemiology. The architecture is further modified to tackle inference in a spatial setting (**Sec.** 5.2-**Fig.** 5.7).

3. Background

We first provide an overview of the key components—score-based diffusion models, transformers, and coupling layers —in this section before giving a detailed description of the methods in **Section** 4.

3.1 Normalizing flow

In Probabilistic Machine Learning, many tasks depend on flexible models for conditional distributions. Although Gaussian distributions are computationally efficient, they often do not provide the flexibility required for more complex data. Consequently, a primary goal in statistics and machine learning is to techniques and theories that enable richer probabilistic models. Normalizing flows (Papamakarios et al., 2021) meet this need by providing a robust framework for defining expressive probability distributions over continuous random variables. They start with a simple base distribution and apply a series of bijective transformations to enhance complexity. At the lowest level of flow-based modeling, we have a D-dimensional vector **x**, and we aim to define a joint distribution over **x**. The core concept is to express **x** as a transformation **f** of a real vector **z** sampled from $p_z(\mathbf{z})$:

$$\mathbf{x} = f(\mathbf{z}) \quad \text{where} \quad \mathbf{z} \sim p_z(\mathbf{z}). \tag{3.1.1}$$

The distribution $p_z(\mathbf{z})$ is known as the base distribution. Both the transformation **f** and the base distribution $p_z(\mathbf{z})$ may have their own parameters, represented by θ and ψ, respectively. This leads to a family of distributions denoted as $p_{\mathbf{x}}(\mathbf{x};\Phi)$, where $\Phi = \{\theta, \psi\}$. A fundamental aspect of flow-based models is that the transformation **f** must be a diffeomorphism, which means it is a differentiable bijection. Additionally, the dimensions of $\mathbf{z}$ and $\mathbf{x}$ must match . Under these conditions, the density of $\mathbf{x}$ can be determined using a change of variables (Rudin et al. (1964); Bogachev and Ruas (2007)).

$$p_x(\mathbf{x}) = \frac{1}{\mid det\ J_{\mathbf{f}}(\mathbf{z}) \mid} p_z(\mathbf{z}) = p_z(\mathbf{f}^{-1}(\mathbf{x})) \mid det\ J_{\mathbf{f}^{-1}}(\mathbf{x}) \mid \quad \text{where} \quad \mathbf{z} = \mathbf{f}^{-1}(\mathbf{x}) \tag{3.1.2}$$

The transformation f can be viewed as reshaping the space $\mathbb{R}^D$ to align the density $p_{\mathbf{x}}(\mathbf{x})$ with $p_z(\mathbf{z})$. The determinant of the Jacobian matrix, $|\det J_{\mathbf{f}}(\mathbf{z})|$, serves as the local scaling factor that quantifies how volumes in D-dimensional space change as a result of f. A notable characteristic of invertible and differentiable transformations is their composability. If there are n such transformations $f_1, \ldots, f_n$, their composition $f_1 \circ f_2 \circ \cdots \circ f_n$ remains both invertible and differentiable, with the inverse and Jacobian determinant defined as follow

$$(f_1 \circ f_2 \circ \cdots \circ f_n)^{-1} = f_n^{-1} \circ f_{n-1}^{-1} \circ \cdots \circ f_1^{-1} \tag{3.1.3}$$

$$det\ J_{f_1 \circ f_2 \circ \cdots \circ f_n}(\mathbf{z}) = \prod_{i=1}^{n} det\ J_{f_i}(\mathbf{z}) \tag{3.1.4}$$

Consequently, complex transformations can be constructed by sequentially composing simpler transformations $f_1, f_2, \ldots, f_n$ to form a composite transformation $f = f_1 \circ f_2 \circ \cdots \circ f_n$. This composition preserves the properties of invertibility and differentiability. Each transformation f_i maps $\mathbf{z}_{i-1}$ to $\mathbf{z}_i$, with $\mathbf{z}_0 = \mathbf{u}$ and $\mathbf{z}_n = \mathbf{x}$. The term "flow" refers to the movement of samples from the base distribution $p_{\mathbf{z}}(\mathbf{z})$ as they undergo these transformations. Conversely, "normalizing" describes the reverse process, where the inverse transformations $f_n^{-1}, \cdots, f_1^{-1}$ convert samples from the target distribution $p_{\mathbf{x}}(\mathbf{x})$ back into the base distribution $p_{\mathbf{z}}(\mathbf{z})$ (Papamakarios et al., 2021), which is often selected as a multivariate normal distribution.

To align the flow-based model $p_{\mathbf{x}}(\mathbf{x};\Phi)$ with a target distribution $p^*_{\mathbf{x}}(\mathbf{x})$, one minimizes the divergence or discrepancy between the two. A common approach involves using *f-divergences* (Wan et al., 2020), which compare distributions through their density ratios.

$$D_f\left[p^*_{\mathbf{x}}(\mathbf{x}) \mid\mid p_{\mathbf{x}}(\mathbf{x};\Phi)\right] = \mathbb{E}_{p_{\mathbf{x}}(\mathbf{x};\Phi)}\left[f\left(\frac{p^*_{\mathbf{x}}(\mathbf{x})}{p_{\mathbf{x}}(\mathbf{x};\Phi)}\right)\right] \quad with\ f\ convex. \tag{3.1.5}$$

In particular, when the function is defined as $f(r) = r\log r$, we obtain the Kullback–Leibler (KL) divergence (Van Erven and Harremos, 2014) , a widely used measure in scenarios where we have samples from the target distribution or can generate them. Thus, the objective can be expressed as:

$$\begin{aligned}\mathcal{L}(\Phi) = D_{KL}\left[p^*_{\mathbf{x}}(\mathbf{x}) \mid\mid p_{\mathbf{x}}(\mathbf{x};\Phi)\right] &= -\mathbb{E}_{p^*_{\mathbf{x}}(\mathbf{x})}\left[\log p_{\mathbf{x}}(\mathbf{x};\Phi)\right] + const.\\ &= -\mathbb{E}_{p^*_{\mathbf{x}}(\mathbf{x})}\left[\log p_{\mathbf{z}}(\mathbf{f}^{-1}(\mathbf{x};\theta);\psi) + \log\left|\det J_{\mathbf{f}^{-1}}(\mathbf{x};\theta)\right|\right] + \text{const.}\end{aligned}$$

Assuming we have a set of samples $\{x_i\}_{i=1}^N$ from the target distribution $p^*_{\mathbf{x}}(\mathbf{x})$, the Monte Carlo approximation of the expectation over $p^*_{\mathbf{x}}$ gives

$$\mathcal{L}(\Phi) \approx -\frac{1}{N}\sum_{i=1}^{N}\log p_{\mathbf{z}}(\mathbf{f}^{-1}(\mathbf{x}_i;\theta);\psi) + \log\left|\det J_{\mathbf{f}^{-1}}(\mathbf{x}_i;\theta)\right| + \text{const.} \tag{3.1.6}$$

Minimizing (3.1.6) is equivalent to fitting the flow-based model to the samples $\{x_i\}_{i=1}^N$ using maximum likelihood estimation.

In practice, the transformation $\mathbf{f} = f_1 \circ f_2 \circ \cdots \circ f_n$ is typically constructed by implementing f_i (or f_i^{-1}) with *Neural Networks*, with $p_z(\mathbf{z})$ chosen as a simple density, such as the multivariate standard normal. Therefore, it is essential to determine how to parameterize $\mathbf{f}_\theta = f_{\theta_1} \circ f_{\theta_2} \circ \cdots \circ f_{\theta_n}$ to allow for the application of the change of variables. Increasing the "depth" of the transformation—defined as the number of composed sub-flows f_{θ_i}—leads to a growth in computational complexity of only $\mathcal{O}(n)$. This enhanced expressivity comes at a practical cost.

Coupling layer For most applications of flow-based models , the Jacobian determinant should be computed in linear time relative to the input dimensionality. Therefore, the flow layer must be designed to be *invertible, differentiable, and have an efficiently computable determinant of the Jacobian* ($\mathcal{O}(D)$ instead of $\mathcal{O}(D^3)$).

Different normalizing flows vary in how they implement these criteria. Autoregressive flows are among the most popular. The key idea is to transform certain parameters based on the values of others, with dependencies modelled by, for example, a neural network (Huang et al., 2020). However, fully masked autoregressive flows exhibit computational asymmetry, which affects their application and usability; either sampling or density evaluation will be D times slower than the other.

A computationally symmetric alternative, which allows for equally fast evaluation and inversion, is the *coupling layer* (Dinh et al., 2016). This method involves selecting an index d (commonly D/2, rounded to the nearest integer) and splitting z into two parts such that $z = [z_{\leq d}, z_{>d}]$. The first part is transformed elementwise independently of the other dimensions, while the second part is transformed elementwise in a manner that depends on the first part.

Considering an affine transformation f, **Fig. A.1** illustrates the forward and reverse workflows of an affine coupling layer, conditioned on side information c, as a combination of scaling s and shifting t, implemented as neural networks.

3.2 Simulation-based inference

Mechanistic models are often implicit, meaning they lack a defined likelihood function $p(y|\theta)$ that describes the relationship between observable variables y and model parameters θ. Instead, these models are presented as simulators that take the parameters θ as input and generate simulated variables y (Diggle and Gratton, 1984). This simulator-based modeling approach is commonly employed in scientific fields such as cosmology (Alsing et al., 2018), high-energy physics (Brehmer et al., 2019), and computational neuroscience (Gonçalves et al., 2020). The process of inferring the parameters θ of a simulator-based model using observed data is often referred to as *likelihood-free inference* (Papamakarios et al., 2019), or *simulation-based inference* (Cranmer et al., 2020).

In a traditional Bayesian framework, we start with a prior $p(\theta)$ and a likelihood $p(y|\theta)$, and we aim to determine the posterior distribution of parameters θ given the observed data y based on Bayes' theorem.

$$p(\theta \mid y) = \frac{p(y \mid \theta)p(\theta)}{p(y)}, \quad p(y) = \int p(y \mid \theta)p(\theta)d\theta. \tag{3.2.1}$$

However, we often cannot analytically compute the posterior in (3.2.1) primarily because the denominator (evidence) involves an intractable integral over the domain of θ. The solution to this problem have been computational algorithms such as Monte Carlo methods(Hammersley, 2013) (including rejection sampling, importance sampling, MCMC, etc.) and Variational Inference (VI) (Blei et al., 2017) to approximate the posterior with a surrogate distribution. The simulation-based inference problem is different; we may not even be able to express the likelihood $p(y \mid \theta)$ analytically, which is common in complex scientific models.Consequently, these latter methods cannot be applied in this situation. Nevertheless, we do have a (stochastic) process that generates data based on a parameter, meaning we can sample from $P(y \mid \theta)$.

$$\theta \longrightarrow Simulator \longrightarrow y.$$

We can ideally generate an unlimited number of simulated data, denoted as y_{sim}, but we only have one observed data point, y_{obs}.The goal, similar to MCMC methods, is to determine the posterior distribution of θ based on the actual empirical data, or, at the very least, to sample from this distribution. Approximate Bayesian Computation (ABC) (Beaumont, 2010) is one of the traditional methods used to address this problem.

Considering the developments in *ABC* methods (see **App.** A.2), one might question why we limit ourselves to the acceptance of only the ϵ-ball (i.e., $\|y_{sim} - y_{obs}\| < \epsilon$). Furthermore, if we had more information about the posterior, we could use it to better constrain the prior in the next round. These two concepts are precisely what Papamakarios and Murray (2016) identified in their Sequential Neural Posterior Estimation (SNPE-A) approach: we do not necessarily need the threshold ϵ or a distance metric ρ. Instead, we can discard these ad-hoc decisions and utilize Neural Density Estimators (NDEs), such as Mixture Density Networks (MDN) (Bishop, 1994). These NDEs are neural networks that output distributions—typically Gaussian Mixture Models (GMM)—trained on y_{sim} by maximizing the log-likelihood. We can then input y_{obs} to obtain the true posterior $p(\theta \mid y_{obs})$, which can optionally serve as the prior for the next round (i.e., Sequential NPE), enhancing both accuracy and efficiency .

When employing the algorithm in a sequential manner, it is essential to account for posterior correction. If we use a proposal distribution $\tilde{p}(\theta)$ instead of the actual prior $p(\theta)$, the NDE posterior q_w becomes $q_w(\theta \mid y) = \frac{\tilde{p}(\theta)p(y|\theta)}{p(y)}$. To obtain the actual posterior, we must make a correction as follows:

$$p(\theta \mid y) = \frac{p(\theta)p(y \mid \theta)}{p(y)} = \frac{p(\theta)}{\tilde{p}(\theta)} q_w(\theta \mid y) \tag{3.2.2}$$

Therefore, to be able to still calculate the posterior analytically, Papamakarios and Murray (2016) restricted the NDE to *GMM*, the proposal to Gaussian, and the prior to Uniform&Gaussian. However, Gonçalves et al. (2020) (SNPE-B) circumvent this problem entirely by putting this correction already in the loss:

$$\mathcal{L}(w) = -\frac{1}{N}\sum_n \frac{p(\theta_n)}{\tilde{p}(\theta_n)} q_w(\theta_n \mid y_n) \tag{3.2.3}$$

They demonstrated that minimizing this (3.2.3) leads to the correct posterior. As a result, there are no longer any constraints, allowing the use of any proposal and more expressive neural density estimators (NDEs), such as those based on Normalizing Flows: Masked Autoregressive Flows (Papamakarios et al., 2017) and Masked Autoencoder for Distribution Estimation (MADE) (Germain et al., 2015). The algorithm was further advanced by Greenberg et al. (2019) with SNPE-C.

When performing inference with Approximate Bayesian Computation for a new set of observed data, it is necessary to repeat most steps in the inference chain, especially if the proposal distribution depends on the observed data. This method scales poorly with a large number of observations. In contrast, inference based on density estimation can amortize, meaning that the computationally expensive steps do not need to be repeated for new observations. This is particularly advantageous in cases with independent and identically distributed (i.i.d.) observations.

3.3 Score-based diffusion models

A diffusion model is a type of generative model that generates data by progressively transforming a simple initial distribution, such as Gaussian noise, into a more complex target distribution that reflects the structure of the data. This model typically involves two processes: forward diffusion and reverse diffusion.

In the forward process, a real data point is gradually corrupted by adding Gaussian noise at each time step, ultimately resulting in pure noise. In the limit of many small time steps, the diffusion process that transforms an initial data distribution $q_0(x_0) = p(x)$ into a simpler distribution $q_T(x_T) = \mathcal{N}(x_T, 0, \mathcal{I})$ is defined as the solution to stochastic differential equations (SDEs), which are commonly expressed as

$$dx_t = f(x_t, t)dt + g(t)dw_t \tag{3.3.1}$$

With w_t being a standard Wiener process, and f and g representing the drift (controlling deterministic properties of the stochastic process) and diffusion coefficients, respectively. Samples from the generative model are then generated by simulating the corresponding reverse diffusion process (Haussmann and Pardoux, 1986) .

$$dx_t = [f(x_t, t) + g(t)^2 \nabla_{x_t} \log q_t(x_t)]dt + g(t)d\tilde{w}_t \tag{3.3.2}$$

This relies on the knowledge of the score function $\nabla_{x_t} \log q_t(x_t)$ at each step.The exact marginal score, as described by (A.3.1) is typically intractable but can be estimated through time conditional denoising score-matching (A.3.2). Given that the conditional score is known, $q_t(x_t \mid x_0) = \mathcal{N}(x_t; \mu_t(x_0), \sigma_t(x_0))$, the score model $s_\theta(x_t, t)$ is trained to minimize the loss

$$\mathcal{L}(\theta) = \underbrace{\mathbb{E}_{t\sim\mathcal{U}(0,T)}}_{\substack{\text{diffusion}\\ \text{time t}}} \underbrace{\mathbb{E}_{x_0\sim q_0(x_0)}}_{\substack{\text{data}\\ \text{sample } x_0}} \underbrace{\mathbb{E}_{x_t\sim q_t(x_t|x_0)}}_{\substack{\text{diffused data}\\ \text{sample } x_t}} [\lambda(t) \parallel \underbrace{s_\theta(x_t, t)}_{\substack{\text{neural}\\ \text{network}}} - \underbrace{\nabla_{x_t} \log q_t(x_t \mid x_0)}_{\substack{\text{score of diffused}\\ \text{data sample}}} \parallel_2^2] \tag{3.3.3}$$

where $\lambda(t)$ denotes a positive weighting function. After expectation, $s_\theta(x_t, t) \approx \nabla_{x_t} \log q_t(x_t)$.

3.4 Transformers

Transformers are a type of neural network architecture primarily used in NLP that overcome the limitations of feed-forward networks in effectively dealing with sequential inputs by incorporating an attention mechanism to capture long-range dependencies in data. They operate by encoding input sequences through three learnable linear projections: *query* (Q), *key* (K), and *value* (V), where *query* represents the current elements, *key* helps compute attention scores, and *value* carries the actual information. The process involves transforming input embeddings into Q, K, and V using learned weight matrices, computing scaled attention scores via the dot product of Q and K , normalizing these scores with *softmax* to obtain attention weights, and finally producing an output as a weighted sum of the values (Vaswani, 2017) followed by feedforward neural networks for processing.

$$Attention(Q,K,V) = Softmax\Big(\frac{QK^T}{\sqrt{d}}\Big).V \tag{3.4.1}$$

4. Methods

We employ two complementary approaches that leverage the capabilities of generative models for inference, aiming to circumvent limitations associated with Bayesian inference using sampling methods like MCMC. The first approach involves training a diffusion model with transformer encoder architectures **Sec.** 4.1 on the joint distribution of hidden variables and data. This model encodes known dependencies between variables, ensuring efficient training of arbitrary conditional distributions. The second method exploits the flexibility of a Bayeflow architecture **Sec.** 4.2, which combines invertible networks with a conditioning module. The structure of this module depends on the inference task, which may involve either a feature extractor sub-network **Sec.** 5.3 or a mask vector **Sec.** 5.1. The model is trained on simulations derived from priors and likelihood during the forward process and allow for fast posterior inference through the reverse transformation.

4.1 Transformer-based diffusion models for spatial inference

We leveraged the capabilities of a probabilistic diffusion model that utilizes transformers to estimate scores. The architecture is inspired by the Simformer model (Gloeckler et al., 2024a), an amortized Bayesian inference method that achieves state-of-the-art performance in simulation-based amortized inference on benchmark tasks (Lueckmann et al. (2021) ; Phillips et al. (2012); Lotka (1927)). Within the simulation-based inference framework, Simformer is trained on the joint distribution of parameters and data and, allowing to exploit knowledge of their conditional dependency structures by encoding them in the attention mask of the transformer (Weilbach et al., 2023). It jointly encodes parameters and data, enabling efficient sampling of arbitrary conditional densities, including posterior and likelihood.

For the spatial inference problem, we used realizations $f = (f(x_1), ..., f(x_n)) \sim \mathcal{GP}(.)$ of Gaussian process priors as input data during the training process **Sec.** 4.1.2. The encoder part of the transformer architecture serves as the time-conditional score model $s_\theta(f_t, t)$ **Sec.** 3.3. It takes as input the embeddings obtained from a specifically designed tokenizer **Sec.** 4.1.1 of the diffused samples f_t and outputs the corresponding estimated $score_t$. Finally, the reverse diffusion process (3.3.2) is used to produce samples given fixed observations. **Fig.** 4.1 illustrates the training architecture and its essential elements.

4.1.1 Embeddings and dependency structures

Prior to being fed into the transformer architecture, the input sequences $f_t = (f_t(x_1), ..., f_t(x_n))$ pass through a tokenizer. This tokenizer represents each variable (token) with three components: an **identifier** that uniquely identifies the variable, a representation of the **value** of the variable, and a **condition state** (Fig. 4.1). The condition state is a binary variable that indicates whether the variable is conditioned upon or not. Together, the condition states of all variables, denoted as $C_M \in \{0, 1\}^n$, form a mask that informs the model about which data points are observed and which are unobserved. Adopting $C_M^{(i)} = 1$ for certain variables in the sequence implies that these variables will not be corrupted during the forward diffusion process, and inference will only be conducted for the other variables (i.e., those with a condition state set to 0), resulting in the training of a conditional diffusion model.

The tokenizer then jointly embeds the values, unique identifiers, and the condition mask into a unified vector representation (embedding) that captures the meanings of the tokens (variables) in a continuous space. For the *value embeddings*, we replicated the value into a vector of a chosen latent dimension, while learnable vector embeddings were used for *identifiers* and *condition states*. Additionally, the diffusion

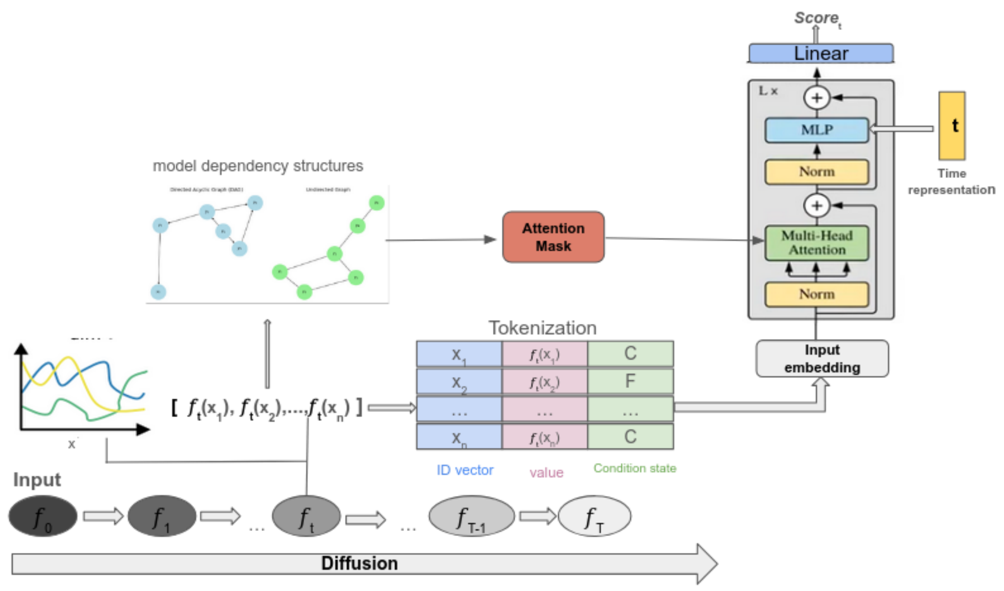

Figure 4.1: Transformer-based diffusion models (Simformer) architecture (Gloeckler et al., 2024a). The diffused realizations of GPs are simplified into a token representation that includes a unique ID, value, and conditional state (free (F) or conditioned (C)). This sequence of tokens is then processed by a transformer model, where variable interactions are managed using an attention mask. The transformer architecture returns a score that is subsequently used to generate samples from the score-based diffusion model.

time was embedded using Fourier transforms to efficiently parameterize the time variable, enabling the transformer (score model) to incorporate temporal information into the score estimates.

We enforce global attention (fully undirected dependencies) among the variables via a dense symmetric attention mask A_M within the attention layers of the transformer, allowing all variables in the sequence $f_t = (f_t(x_1), ..., f_t(x_n))$ to attend to each other.

4.1.2 Training and inference

The Transformer takes as input the result embedding from the tokenizer and incorporates the defined attention mask A_M in the attention layers to produce the score . The latter is optimized using denoising score matching (A.3). The condition mask C_M can be fixed during training if targeting a specific conditional, it prevents noise from being added to the conditioned variables ($C_M^{(i)} = 1$). At noise level t, the forward diffusion generates a partially noisy sample described by $f_t^{C_M} = (1 - C_M) \cdot f_t + C_M \cdot f_0$, ensuring that the variables we want to condition on remain clean. The objective defined in equation (3.3.3) is then stated as:

$$\mathcal{L}(\theta) = \mathbb{E}_{C_M,t,f_0,f_t}\Big[\parallel (1 - C_M). \Big(s_\theta^{A_M}(f_t^{C_M}, t) - \nabla_{f_t} \log q_t(f_t \mid f_0)\Big) \parallel_2^2 \Big] \tag{4.1.1}$$

where $s_\theta^{A_M}$ denotes the score model equipped with a specific attention mask A_M.

After training the Simformer, we draw samples from the noise distribution and apply the reverse diffusion process (3.3.2) to all unobserved variables, while keeping the observed variables constant at their conditioning value.

Training algorithms:

Algorithm 1 Simformer Training

```
1: Set attention mask $A_M$ ( causal relation between variables in the sequence )
2: repeat
3:     set condition mask $C_M$
4:     draw sample  $f_0 \sim \mathcal{GP}(.)$
5:     $t \sim Uniform(\{1, ..., T\})$  (sample diffusion time)
6:     diffuse sample $f_t \sim q_t(f_t \mid f_0)$  (forward diffusion )
7:     Take gradient descent step $\nabla_\theta$ $\| (1 - C_M).(s_\theta^{A_M}(f_t^{C_M}, t) - \nabla_{f_t} \log q_t(f_t \mid f_0)) \|_2^2$
8: until Convergence ( =0
```

Algorithm 2 Simformer Inference

```
1: Initialize $t \leftarrow T$ , $y \sim q_T(y) \approx \mathcal{N}(0, I)$  ( draw from the noise distribution )
2: repeat
3:     $\Delta y \leftarrow \left[f(y,t) - g^2(t) S_\theta(y^{C_M}, t)\right] \Delta t + g(t) z$ , $z \sim \mathcal{N}(0, |\Delta t| I)$
       ( Euler-Maruyama numerical solver )
4:     $y \leftarrow y + \Delta t$
5:     $t \leftarrow t + \Delta t$
6: until t =0
```

4.2 BayesFlow approach with conditional Invertible Neural Networks

This approach follows the workflow of the simulation-based inference framework for both inference on stochastic models and spatial inference problem. We designed an instantiation of a flow-based architecture (**Sec.** 3.1) that comprises two sub-networks: a conditioning module g_η for feature extraction or filtering and an invertible inference network that performs Bayesian parameter inference based on the output feature vector. **Fig.** 4.2 illustrates the training and inference phases of our architecture. The invertible network f_ϕ is tasked with inverting the forward model using the information extracted

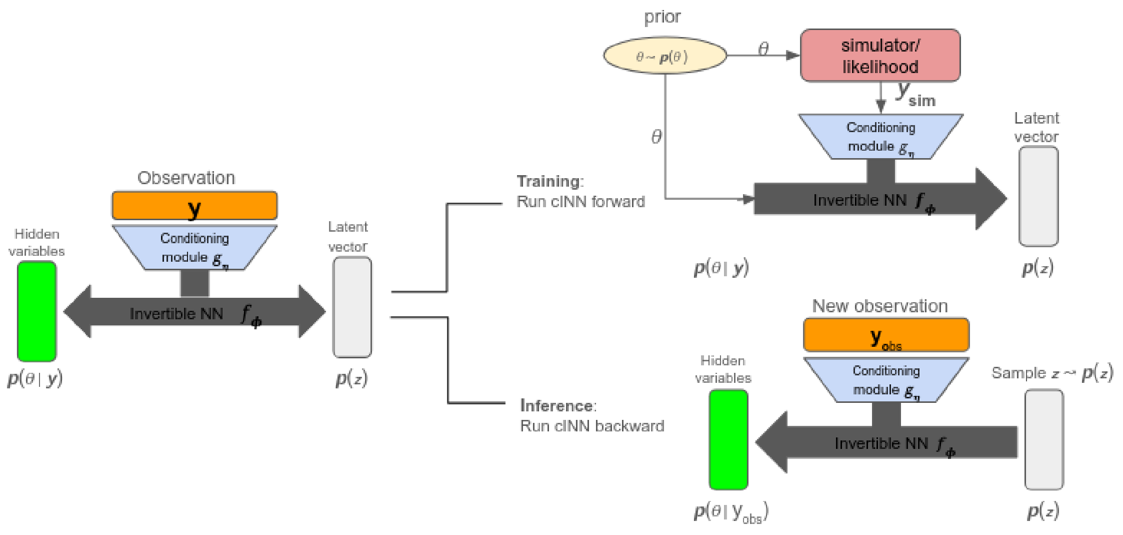

Figure 4.2: Structure and workflow of the cINN. During the training phase (forward direction), the implicit likelihood of the assumed model (e.g., SIR model) is used to simulate data based on the prior distributions of the hidden variables θ. The synthetic data (θ, y_{sim}) are then used to train the composite neural network, which consists of a conditioning module g_η (feature extractor **Sec.** 5.3 or mask vector **Sec.** 5.1) and an inference network f_ϕ. During the inference phase (backward evaluation), the real data y_{obs} is passed to the trained network to infer the posterior distribution of the hidden variables.

by the conditioning module g_η. It was built by concatenating a series of affine coupling layers f_k, alternating with a permutation layer Q to shuffle the coordinates between coupling layers, ensuring that the skipped coordinates do not consistently represent the same set of parameters.

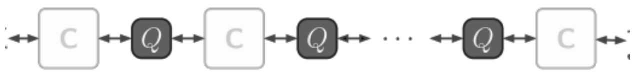

Figure 4.3: INN structure $f_\phi = f_1 o f_2 o ... o f_K$. Stacking a sequence of conditional coupling layers C into a deep invertible neural network , alternating with a permutation (orthogonal) layer Q.

The structure of the conditioning module g_η varies depending on the inference task. For inference on Gaussian processes (**Sec.** 5.1, 5.2) we design g_η as a mask that filters out unobserved data points, passing only the vector of observed points to the inference module. This approach informs the invertible network about the observed points, allowing it to infer the overall patterns based on this information.

For inference on the SIR model (**Sec.** 5.3), which involves noisy time series, g_η consists of 1D convolutional layers for noise reduction and feature extraction, followed by a recurrent summary network (LSTM). The convolutional module extracts relevant features from the time series data while preserving its temporal structure by applying adjustable one-dimensional filters of varying sizes at each level. The intuition behind this design is that filters of different sizes can capture patterns at various temporal scales (e.g., a filter of size one captures daily fluctuations, while a filter of size seven captures weekly dynamics). Its output is a multivariate sequence containing the filtered epidemiological time series. To reduce this filtered sequence to a fixed-size vector, we pass it through a long-short term memory (LSTM) recurrent network (Radev et al., 2021).

4.2.1 Training and inference

The invertible network operates in two modes. During training, the network is evaluated only in the forward direction and is guided by an appropriate optimization criterion to transform the posterior $p(\theta \mid y)$ into a simple base distribution, typically Gaussian, from which samples can be easily drawn. This process allows the inference network to incorporate information from both the prior and the data-generating mechanism (i.e., the implicit likelihood). The estimated posterior $\hat{p}(\theta|y)$ is expressed using the change-of-variable formula as follow

$$z = f_\phi(\theta, g_\eta(y)) \quad s.t. \;\; p(z) = \mathcal{N}(0, \mathbb{I}) \iff \hat{p}(\theta|y) = p_z\left(z = f_\phi(\theta, g_\eta(y))\right) \mid \nabla_\phi f_\phi(\theta, g_\eta(y)) \mid$$

Trained with the maximum likelihood loss (has an especially simple form when $p(z)$ is standard normal)::

$$\begin{aligned}\mathcal{L}(\phi, \eta) = -\log \hat{p}(\theta|y) &= -\log p_z\left(z = f_\phi\left(\theta, g_\eta(y)\right)\right) - \log \mid \nabla_\phi f_\phi\left(\theta, g_\eta(y)\right) \mid \\ &= \frac{D}{2}\log(2\pi) + \frac{1}{2}\left\|f_\phi\left(\theta, g_\eta(y)\right)\right\|_2^2 - \sum_{l=1}^{L} sum\left(\log S_{\phi,l}\left(\theta_{l2}, g_\eta(y)\right)\right)\end{aligned}$$

where $S_{\phi,l}\left(\theta_{l2}, g_\eta(y)\right)$ is the multiplier at coupling layer l.

Algorithm 3 cINN Training

```
1: repeat
2:     Simulate parameters $\theta \sim p(\theta)$
3:     Run the simulation $y_{sim} \sim \mathcal{L}ikelihood(\theta)$
4:     Perform maximum likelihood training with batch of $(\theta, y_{sim})$ pairs
5: until Convergence ( =0
```

During inference, the inference network is evaluated in the reverse direction, utilizing conditional information from real observed data processed through the filtering module. The posterior is approximated by repeatedly sampling from a simple base distribution and applying the inverse of the forward transformation learned during the training phase. Since f_ϕ is invertible, we obtain the generative model at no additional cost when given an observation y_{obs} (observed data points for GP or time series $\hat{y}_{1:T}$ for SIR).

$$\theta \sim \hat{p}(\theta \mid y_{obs}) \quad \Leftrightarrow \quad z \sim \mathcal{N}(0, \mathbb{I}) \;\; and \; \theta = f_\phi^{-1}\left(z; g_\eta(y_{obs})\right). \tag{4.2.1}$$

With the methodological framework in place, we now shift our focus to the empirical validation of the approaches. In the following section, we outline the experimental setup and present the results from applying these methods to synthetic datasets.

5. Experiments

In this section, we present four distinct experiments designed to evaluate the capabilities of the proposed methods. Two of these experiments focus on spatial inference (**Sec.** 5.1, 5.2), aiming to determine the spatial patterns and relationships within the data using Gaussian process priors. The other two experiments concentrate on compartment models in epidemiology (**Sec.** 5.3, 5.4), which aim to infer hidden parameters that drive dynamic behaviors. All results involving diffusion models are obtained using the Variance Exploding SDE (VESDE) (Yang et al.). For each task, synthetic ground truths was simulated. These experiments provide comprehensive insights into the performance and applicability of the methods across various similar tasks.

5.1 One-dimensional Gaussian process inference

In this first example, we employ both approaches (**Sec.** 4.1, 4.2) to perform inference on continuous data $\{y_i\}_{i=1}^n$, on a regular one-dimensional grid of $n = 80$ points over the interval $[0, 1]$. Training prior samples are drawn from a Gaussian Process (GP) with a zero mean and a standardized radial basis function (RBF) kernel, defined as:

$$k(x, x') = \sigma^2 \exp\left(-\frac{\|x - x'\|^2}{2l^2}\right),$$

We set $\sigma = 1$ and impose a uniform hyperprior $\mathcal{U}$ on the lengthscale l. To perform inference, we generate one realization of the GP with $l = 0.2$, which serves as the ground truth, and simulate observed data by adding i.i.d. noise. We vary the number of observed data points to 0.6% (5 data points), 7% (6 data points), and 1% (8 data points) of the total number of grid points, aiming to recover the true function. The model used for inference has the following hierarchical structure:

$$\begin{aligned}
&l \sim \mathcal{U}(0.01, 1) \\
&f \sim \mathcal{GP}\left(0, k_l^{RBF}(.,.)\right) \\
&y \sim \mathcal{N}\left(f, s^2\right), \quad s \sim \mathcal{N}^+(0.2)
\end{aligned}$$

We also examine how well the inference approaches perform under model mis-specification by training on realizations of Gaussian Processes (GPs) using *Matérn Kernel* ($\nu = 2.5$) :

$$k_{\text{Matérn}}(x, x') = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}d}{\ell}\right)^{\nu} K_\nu\left(\frac{\sqrt{2\nu}d}{\ell}\right) \tag{5.1.1}$$

where: $d = \|x - x'\|$ is the Euclidean distance between the two points, ℓ is the length scale, ν is the smoothness parameter, σ^2 is the variance, K_ν is the modified Bessel function of the second kind, $\Gamma(\nu)$ is the gamma function.

Simformer inference

The diffusion-based transformer architecture is trained with a tokenizer that concatenates a 20-dimensional value embedding, a 20-dimensional node ID embedding, and a 10-dimensional condition embedding.

This architecture processes the concatenated tokens using 2 attention heads, each with an attention size of 10, across 2 layers, and applies a widening factor of 10. The training is conducted over 15 epochs on 20,000 realizations $f \sim \mathcal{GP}(.,.)$ of GP priors , with each epoch containing an inner loop of 5,000 iterations. During training, random masks are applied, enabling the model to perform inference under any conditional scenario. After training, we run inference with 10000 posterior samples via reverse diffusion **Alg.** 2 for each set of observed data points.

Fig. 5.1 displays the inference results for three sets of observed data points after training the model using GPs with RBF kernels. As expected, all posterior samples pass through the observed points while effectively capturing the underlying data pattern. The uncertainty region is smooth and diminishes as the number of observed points increases, indicating that the model improves with more data. However, the uncertainty becomes overly compressed between two observed data points, limiting the model's robustness to unexpected effects. Additionally, a few noisy samples fall outside the Bayesian credible interval, although this occurrence is rare. Both observations may be linked to the chosen range of length scale for the kernel, which directly influences the nature of the samples used during training.

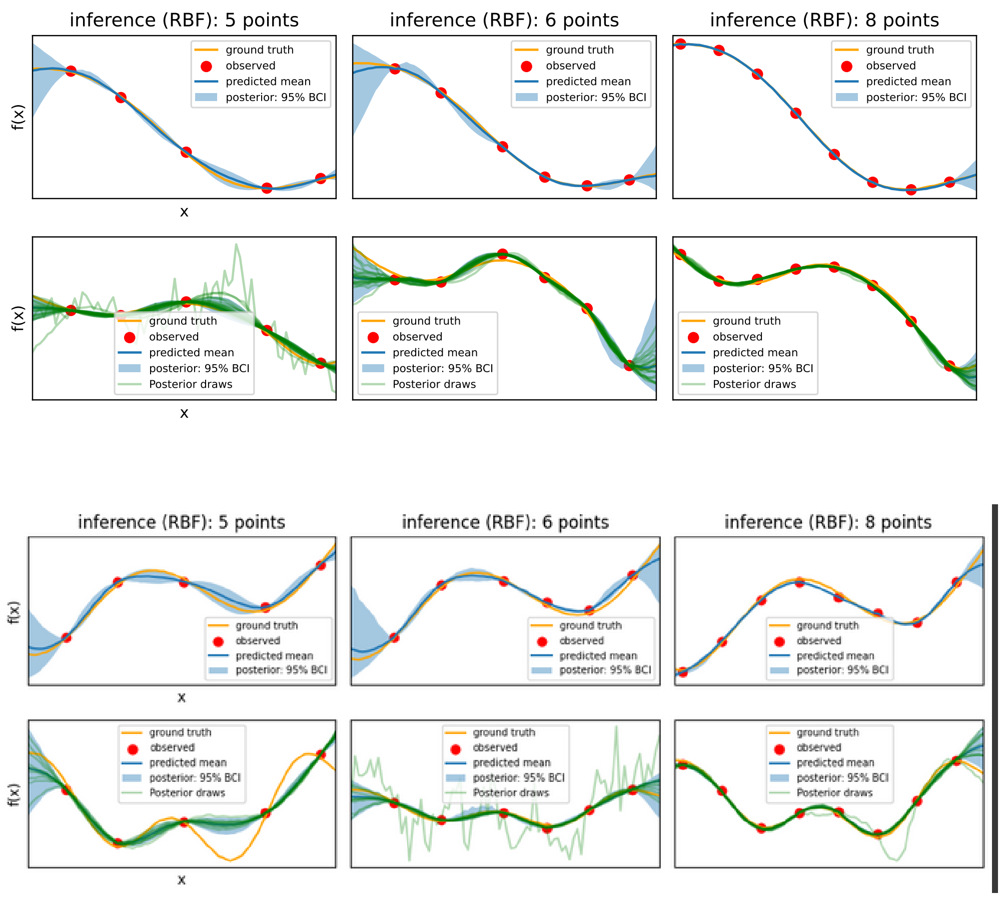

Figure 5.1: Inference results of the Simformer **Sec.** 4.1 on a one-dimensional grid using RBF-based GP priors. Each row represents a separate example, with inference conducted using 6%, 7%, and 10% of the total number of points, respectively. Posterior draws are shown in green, the posterior mean is in blue, and the shaded areas represent the 95% Bayesian credible interval.

In contrast, when employing the Matérn kernel (Fig. 5.2), the results demonstrate a more robust profile with a smoother uncertainty shape and a more reliable credible interval. This suggests that the origin of training samples significantly affects the posterior uncertainty behavior during inference.

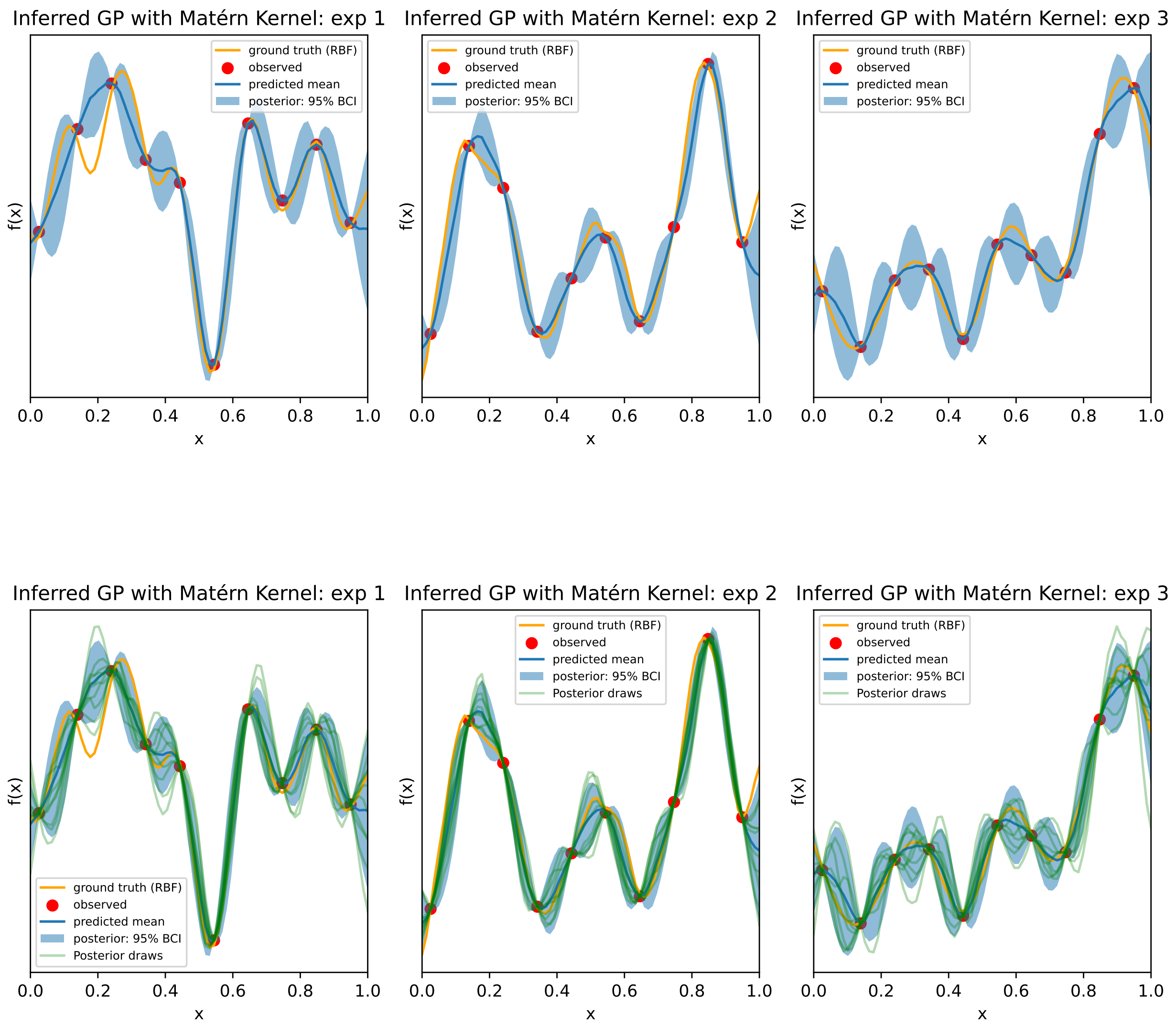

Figure 5.2: Inference results of the Simformer **Sec.** 4.1 on a one-dimensional grid using Matérn kernel-based Gaussian Process (GP) priors. Each column represents a separate example. Inference was conducted using 10% of the total points. The posterior draws are displayed in green, the posterior mean is indicated in blue, and the shaded areas represent the 95% Bayesian credible interval.

Bayesflow inference

The Conditional Invertible Neural Network (cINN) is trained for 8 rounds using a sequential approach. Each round involves simulating 20,000 synthetic data pairs (f_{GP}, y_{sim}), where $y_{sim} \sim \mathcal{N}\left(f_{GP}, s^2\right)$) conditioned on the observed data points $y = g_\eta(y_{sim})$, extracted with the fixed mask g_η , and training for 1000 epochs. The model is parameterized by 10 coupling layers, alternating with permutation layers. After each round, the posterior learned by the cINN is used as the proposal distribution for the next round of simulations. The total runtime is recorded for performance evaluation, with visualizations of the inferred mean over 10000 generated samples to assess the posterior predictive at each stage (Fig. 5.3.

The cINN model exhibits robust performance in rapidly identifying overall trends with minimal observed data, achieving fast convergence within just five sequential training rounds (Fig. 5.3)). As training progresses, uncertainty gradually decreases, demonstrating the model's ability to effectively align the posterior space with the conditioning data. Both training and inference processes are highly efficient (Fig. **Tab.** A.1) and scalable with respect to data dimensionality. However, the model struggles with smoothness, producing noisy uncertainty estimates, which tend to stabilize as more observations are added (Fig. 5.4). While the model's sequential nature limits its adaptability to new observations, we can leverage its rapid training and inference capabilities to efficiently rerun the model for each query.

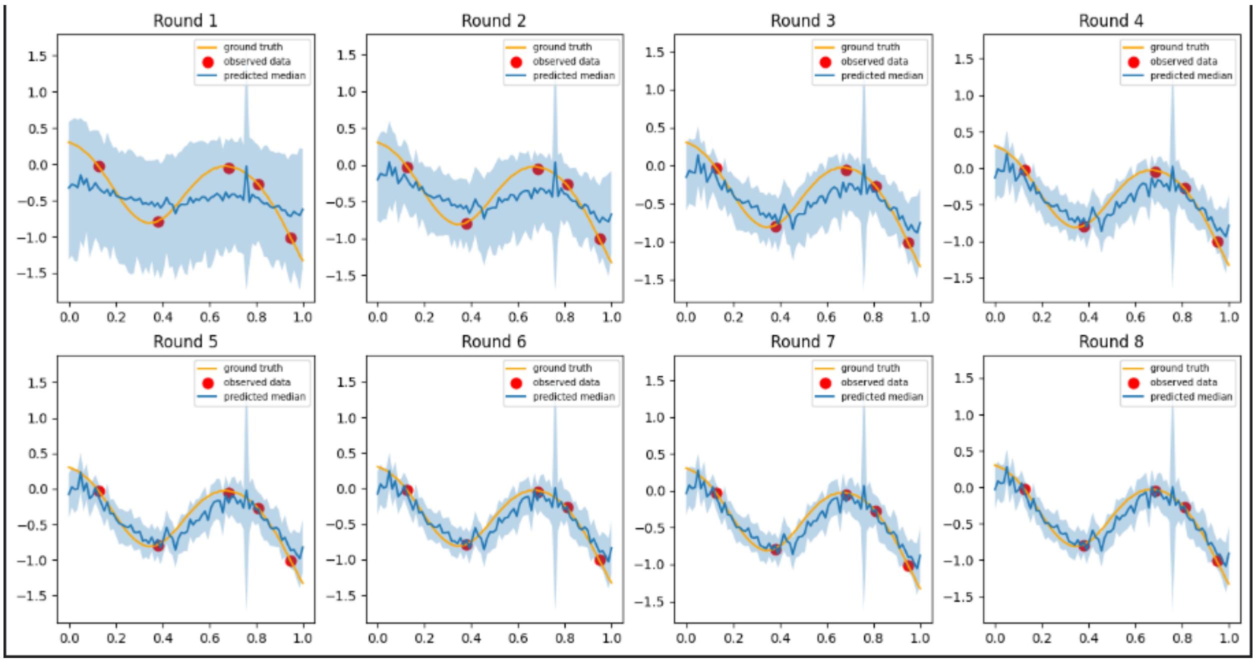

Figure 5.3: Inference results from the flow-based model (cINN) 4.2 using RBF-based GP priors. The model was sequentially trained over 8 rounds, with the posterior from each round serving as the prior for the next. The posterior mean (in blue) and the 95% Bayesian credible interval (represented by shaded areas) were inferred based on 5 observed points at each round. The model ceased updating the posterior after 5 rounds, resulting in a noisy curve that reflected the true pattern.

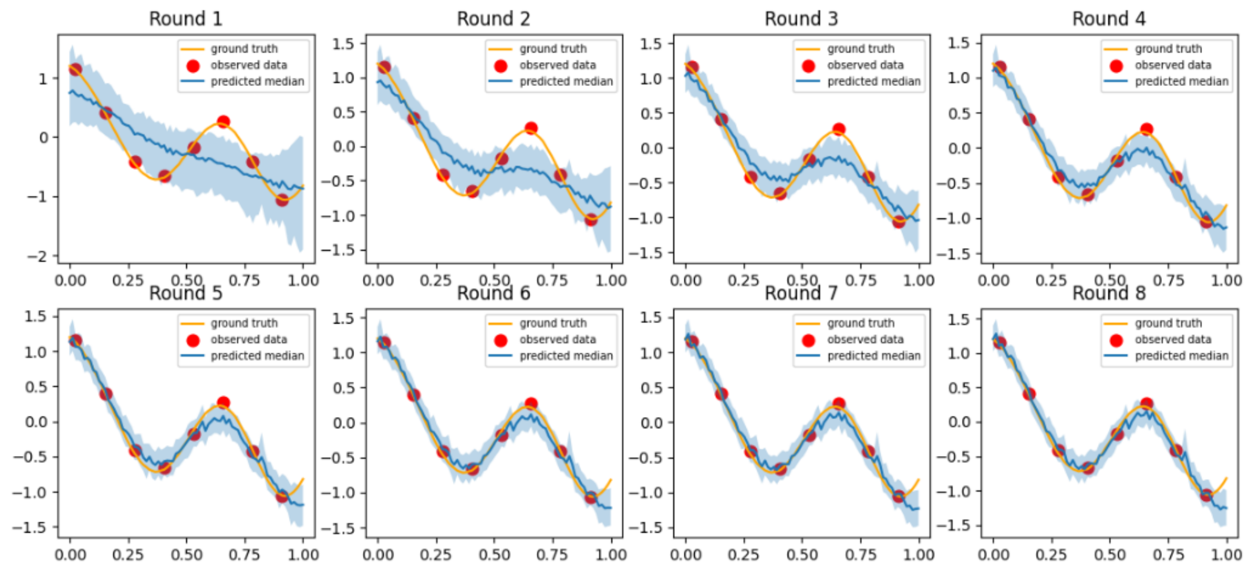

Figure 5.4: Inference results from the flow-based model (cINN) **Sec.** 4.2 using RBF-based GP priors. The model underwent sequential training over 8 rounds, with the posterior from each round functioning as the prior for the subsequent one. The posterior mean is shown in blue, inferred using 8 observed points at each round. The shaded areas represent the 95% BCI. The model stopped updating the posterior after 5 rounds and produced a more stable curve that accurately captured the ground truth pattern.

5.2 Two-dimensional Gaussian process inference

An analogous set of experiments was conducted for two-dimensional Gaussian process priors. A uniform grid was created over a unit square by dividing each axis into 15 segments, resulting in a total of grid cells (N = 225). The training relies on realizations of Gaussian process priors using the Matérn kernel, while the ground truth is based on a GP with the RBF kernel $(l = 0.2)$ (5.1.1) .

Both approaches show improved mean prediction quality and reduced uncertainty in surface estimates as the number of observed data points increases. However, for the diffusion-based model, the standard deviation is less smooth than expected (**Fig.** 5.6).

While both methods effectively learn the spatial correlation between observed data points to reconstruct the surface, the diffusion-based approach can generalize to new observations at locations that align with the conditioning masks applied during training. In contrast, the flow-based model requires re-running the inference process for each new observation. However, this is manageable because of its efficiency, as demonstrated by the execution times in the table **Tab.** A.1.

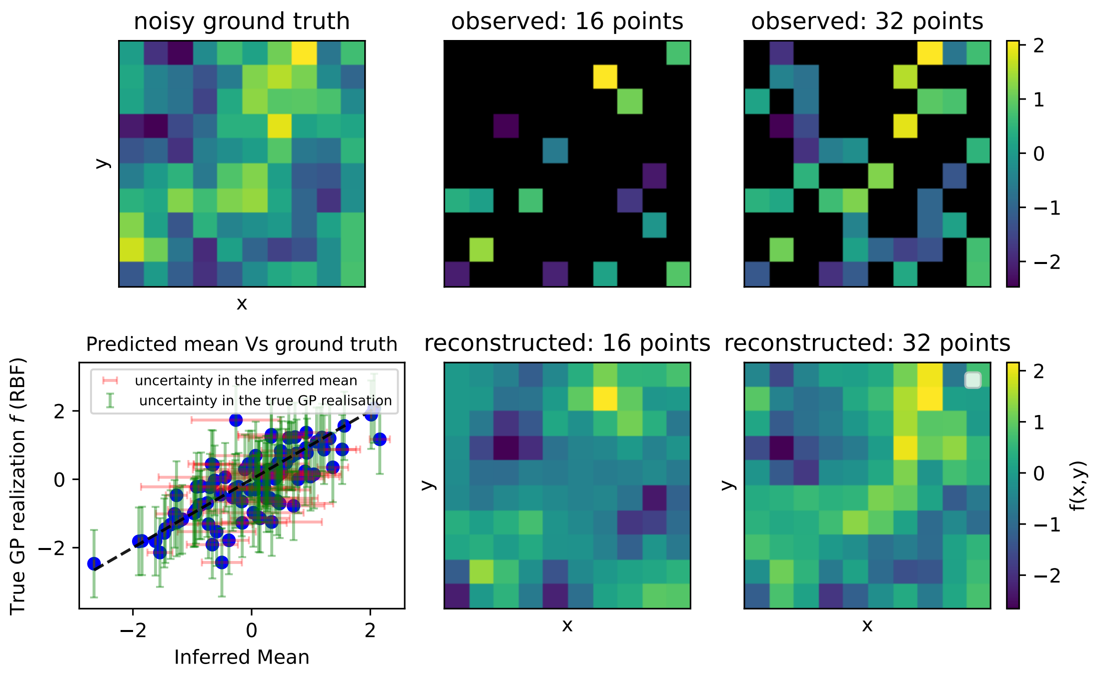

Figure 5.5: Inference results of the Simformer 4.1 on a two-dimensional grid with RBF-based GP priors. Inference was performed using 7% and 14% of the total grid cells (n=225). Colored cells represent the values at observed locations, while black areas indicate unseen (masked) values. The scatter plot evaluates how closely the reconstructed surface (based on 32 observations) aligns with the ground truth. A perfect prediction would follow the black dashed line ($y = x$).

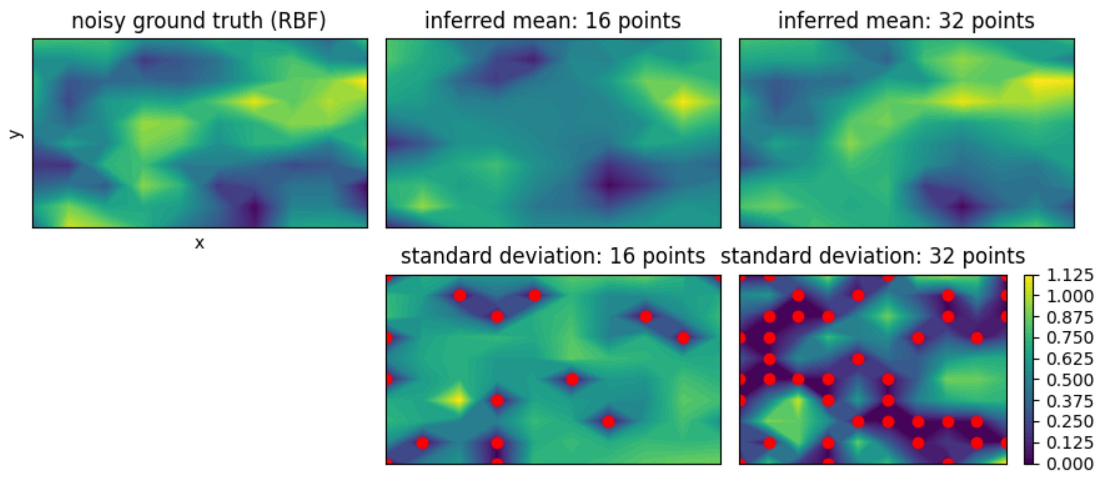

Figure 5.6: Inference results of the Simformer model 4.1 on a two-dimensional grid with Matérn kernel-based GP priors. Inference was performed using 4% and 8% of the total values in the area (n=400). The red scatter points represent the locations of observed values and the second row illustrates the uncertainty in the estimated surfaces.

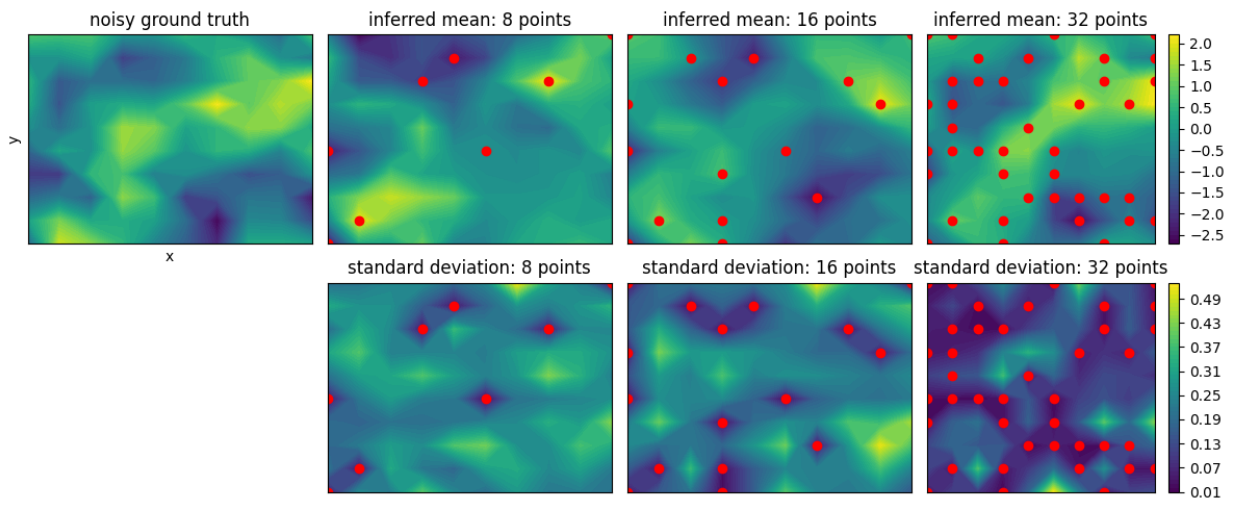

Figure 5.7: Inference results on a two-dimensional grid using the flow-based model (cINN) 4.2 with Matérn-kernel-based Gaussian Process (GP) priors. For comparison, inference was conducted using 2%, 4%, and 8% of the total number of points, respectively. As the number of observed points increases, the uncertainty in the surface estimates decreases.

5.3 Inference for an SIR model

In this section, we demonstrate the ability of our flow-based architecture (Sec. 4.2) in inferring parameters of an epidemiological model with sparse population time series data—SIR model. The SIR model (Kermack and McKendrick, 1927) is a classical mathematical model in epidemiology that describes the spread of infectious diseases within a population. It divides the population into three compartments: Susceptible S, Infectious I, and Recovered R. The dynamics of the model are governed by the following differential equations that describe the transitions between these compartments over time.

$$\frac{dS}{dt} = -\beta\frac{SI}{N} \quad , \quad \frac{dI}{dt} = \beta\frac{SI}{N} - \gamma I \quad , \quad \frac{dR}{dt} = \gamma I \tag{5.3.1}$$

where:

- $S(t)$, $I(t)$, and $R(t)$ represent the number of susceptible, infectious, and recovered individuals at time t.
- N is the total population, with $S + I + R = N$.
- β is the transmission rate representing the likelihood of disease spread through contact between a susceptible and an infectious individual.
- γ is the recovery rate, representing the rate at which infectious individuals recover and become immune.

We simulated 14 observed data points representing the daily reported noisy number of infected individuals. The inference model use $I(t)$ from the SIR model in conjunction with an observation model $y(t)$ (count distribution) that accounts for the discrepancy between true and observed cases. The observation model is defined as y(t) $= \mathcal{N}egBin(I(t), \phi)$, where ϕ is the overdispersion parameter. Let $\theta = (\beta, \gamma, \phi)$ denote the vector of unknown parameters. We use the following priors over θ:

$\beta \sim \mathcal{N}^+(2,1)$, $\gamma \sim \mathcal{N}^+(0.2, 0.5)$, $\phi \sim \exp(6)$, with initial conditions (S_0, I_0, R_0)= (N - 1, 1, 0) where N = 1000.

Inference

We train the conditional invertible network on 5000 simulated synthetic pairs (θ, y) in one round to learn the underlying posterior distribution. The invertible network consists of 8 coupling layers, and the conditioning network incorporates 3 1D Conv layers to capture local patterns while preserving the temporal structure, followed by 1 LSTM layer that summarizes the resulting temporal features into a fixed-size vector, which is used to condition the invertible module and guide the transformation process.

The result model is applied to the observed reported cases, and standard diagnostic tests were conducted to evaluate the reliability of the posterior estimates. The observed and predicted dynamics, as well as the marginal posterior distributions of the model parameters, are illustrated in **Fig.** 5.9 and **Fig.** 5.8 respectively. The posterior predictions closely align with the actual observations, featuring well-calibrated uncertainty bounds. Despite the number of unknown model parameters and limited data, our analysis indicated a significant reduction in uncertainty compared to prior knowledge for all the model parameter (**Fig.** 5.10). Furthermore, the model effectively recovered the true parameters that accurately represent the ground truth dynamics. However, the SBC plots (**Fig.** 5.11) indicate minor systematic biases in the approximate posteriors, which may stem from the choice of priors.

After the training phase, making inference queries for different observations $y_{1:T}$ becomes cost effective, as the training effort quickly amortizes over multiple queries. Furthermore, the length of the time series is not fixed during training or inference, providing flexibility in handling varying time series lengths. The model can also be applied to multivariate epidemiological time series.

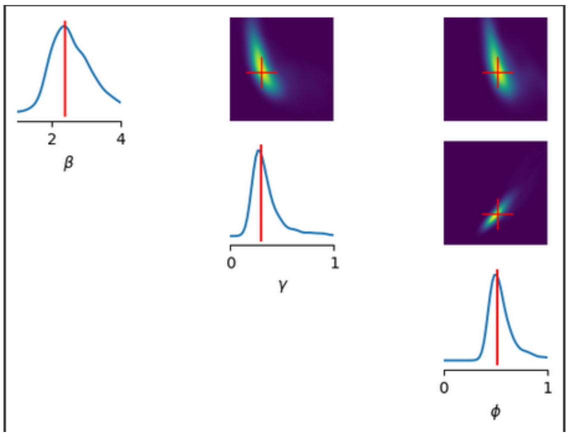

Figure 5.8: Marginal posteriors analysis. We generated 10,000 posterior samples to investigate the estimation (kde) of the marginal distributions. The true parameters, indicated by red lines, lie within the region of highest density in the marginal posterior distributions.

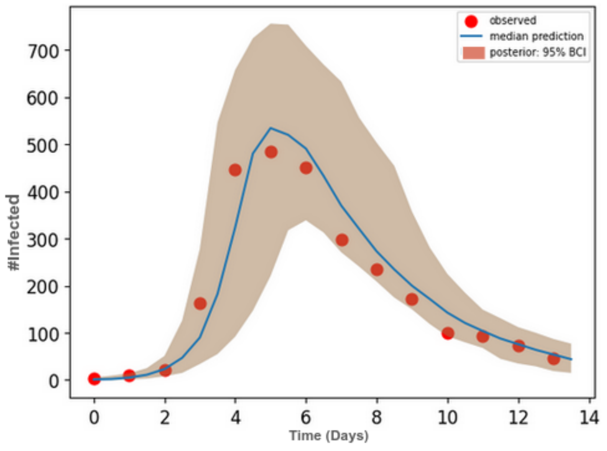

Figure 5.9: Posterior predictions. The mean predicted trajectory of the infected population, $I(t)$, is shown in blue, along with the 95% Bayesian credible interval, represented by the shaded area. The prediction aligns with the observations.

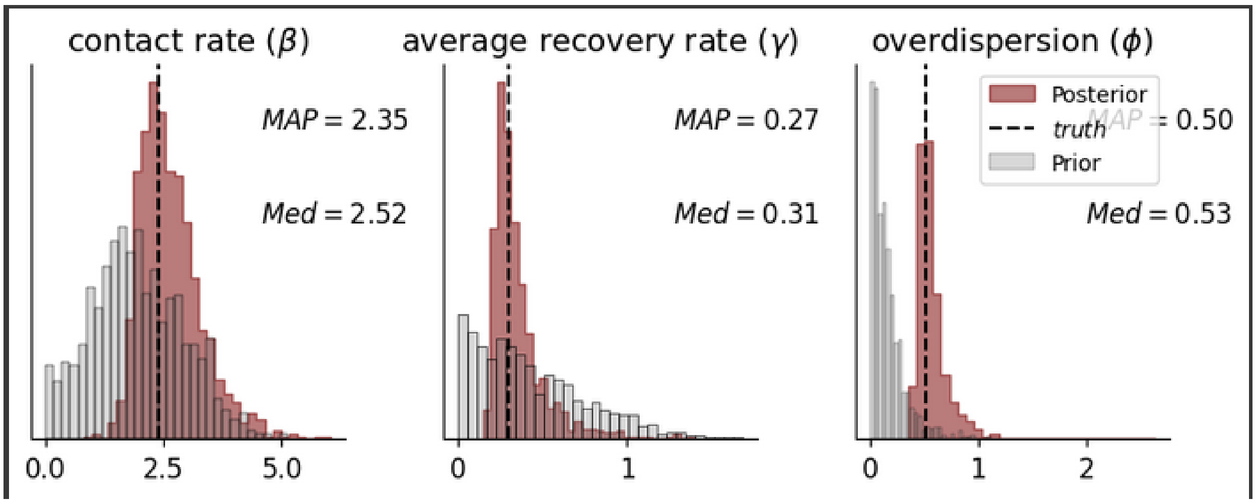

Figure 5.10: Marginal posteriors of the three model parameters inferred from observed data, along with the median and MAP summary statistics. Gray lines represent the prior distributions for comparison, and vertical dashed lines mark the ground truth parameter values.

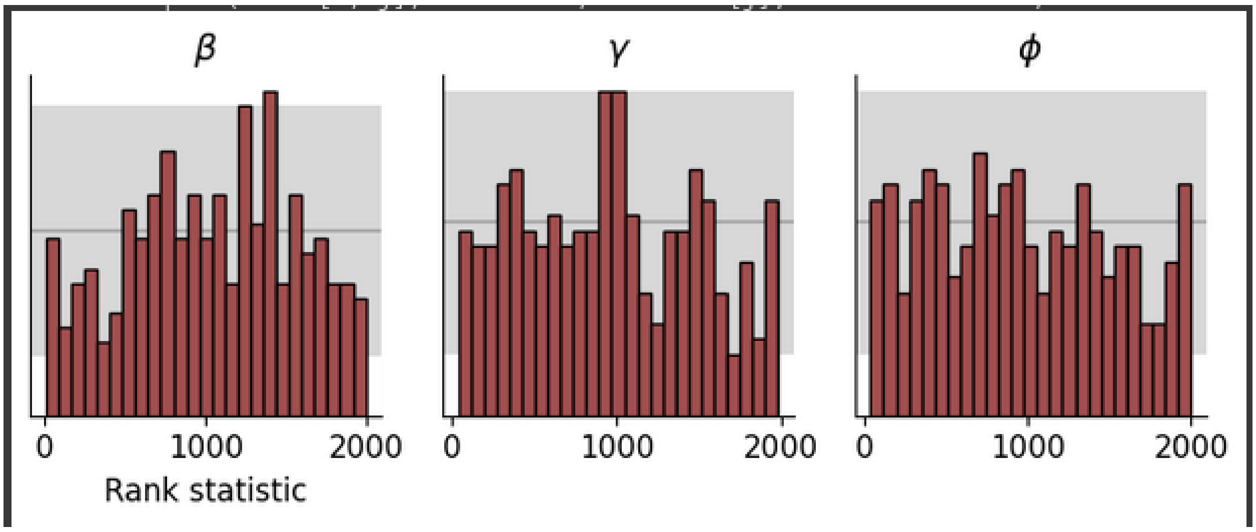

Figure 5.11: Simulation-based calibration (SBC) (Talts et al., 2018) was computed from 10,000 simulated epidemic scenarios. Trustworthy approximation should result in the uniformity of the rank statistics.

In this experiment, the model parameters are assumed to be constant over time, which simplifies the inference task. However, treating the parameters as time-dependent allows for the incorporation of factors such as social distancing measures, public health interventions, and changes in human behavior (Chen et al., 2020) which are essential for accurately simulating the real-world dynamics of disease spread. The next experiment illustrates the capability of the Transformer-based diffusion approach to address a similar inference task in a more complex scenario that involves time-dependent parameters.

5.4 Inference in infinite dimensional parameters: SEIR-model

At its core, the compartmental infectious disease transmission model follows a Susceptible-Exposed-Infected-Removed (SEIR) structure (Fig 1). We extended this model by allowing the transmission rate to vary over time. The model is defined as follows:

$$\frac{dS}{dt} = -\beta\rho(t)cS\frac{I}{N}\ , \quad \frac{dE}{dt} = \beta\rho(t)cS\frac{I}{N} - \tau E\ , \quad \frac{dI}{dt} = \tau E - \gamma I, \quad \frac{dR}{dt} = \gamma I \tag{5.4.1}$$

where $\rho(t)$ represents the time-dependent factor that indicates changes in the transmission rate over time relative to a baseline. β is the probability of a transmission event upon contact, τ is the inverse of the average latent period, γ is the inverse of the average time an individual spends in the infected compartment (i.e., recovery rate), and c is the average number of contacts an individual has per day.

The ground truth data was generated using the approach outlined in Bouman et al. (2024), which simulates an epidemic of a respiratory pathogen with time-varying transmission modeled by a B-spline. The B-spline is uniquely defined by the degree of the polynomial and a set of predefined knots.

We simulated laboratory-confirmed cases over a period of 40 weeks, involving a cohort of 100,000 individuals. The transmission process was modeled with a 25% probability of transmission per contact (β = 0.25), a baseline of 14 contacts per day, and a time-varying component using a cubic spline (degree 3), with knots placed every 4 weeks. We then randomly subsampled 25% of confirmed cases (10 cases) to simulate the scenario of limited available observations. The observational data was modeled with Negative Binomial noise, characterized by a mean of I(t) and an overdispersion parameter of $\phi =$ 0.54 : y(t) $= \mathcal{N}egBin(I(t), \phi)$.

Inference We imposed a GP prior over the time-varying transmission rate, $\rho(t) \sim \mathcal{GP}(0, k_{RBF})$, where k_{RBF} is the RBF kernel With $l = 5$ $(lengthscale)$. We first trained the Transformer-based diffusion architecture (**Sec.** 4.1) on 20,000 joint pairs $(\rho(t), y)_n$ of synthetic simulations derived from the prior and likelihood, while keeping the time-invariant parameter fixed (**Fig.** 5.12). We then trained a second model on the joint $(\beta, \gamma, \tau, \rho(t), y)$ to infer both time-dependent and time-invariant parameters (**Fig.** 5.13). During training, two condition masks were applied, targeting posterior and likelihood conditionals, respectively.

The inference results are illustrated in **Fig.** 5.12. The model successfully recovered a time-varying transmission rate that aligns with the ground truth observations (**Fig.** 5.12). As the number of observed infections decreased, the estimated posterior for the transmission rate exhibited increasing uncertainty. This outcome is expected since conclusive insights about the transmission rate are difficult to obtain in scenarios with few confirmed infections (see **Fig.** A.2). When time-invariant parameters come into play (**Fig.** 5.13), the model successfully recovers most of them (ground truth) while maintaining its accuracy in inferring time-varying transmission, albeit with increased uncertainty in the posterior predictions. Additionally, the architecture's ability to sample any conditional distribution imposed during training enabled us to generate 10,000 posterior predictive samples without needing to run the simulator. These samples closely matched the observed data, further validating the model's accuracy.

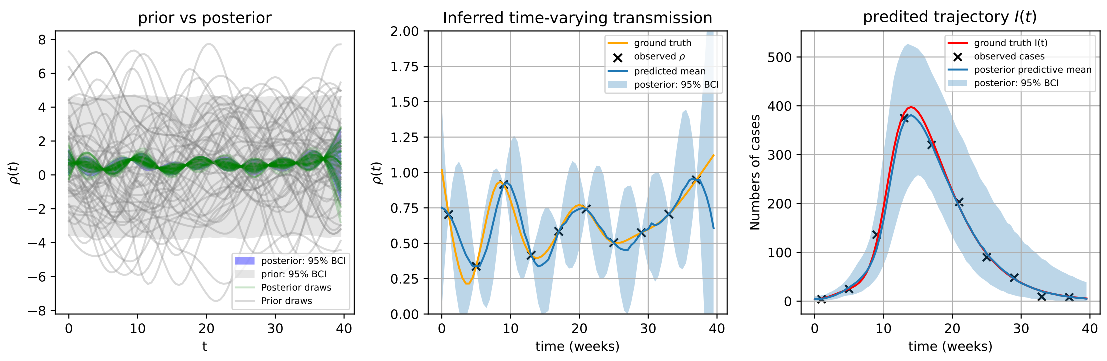

Figure 5.12: Inferred time-varying transmission $\rho(t)$ and the posterior predictive plot (right). The plus signs in the right plot represent sparse (weekly) counts of laboratory-confirmed cases associated with the corresponding transmission rates (middle).The time-invariant parameters were set to their true values, allowing inference to focus only on $\rho(t)$.The left plot compares GP priors with the posterior samples. The inferred function in blue (middle), derived from 10 confirmed cases, closely aligns with the ground truth $\rho(t)$, resulting in an excellent fit, as shown by the posterior predictions on the right.

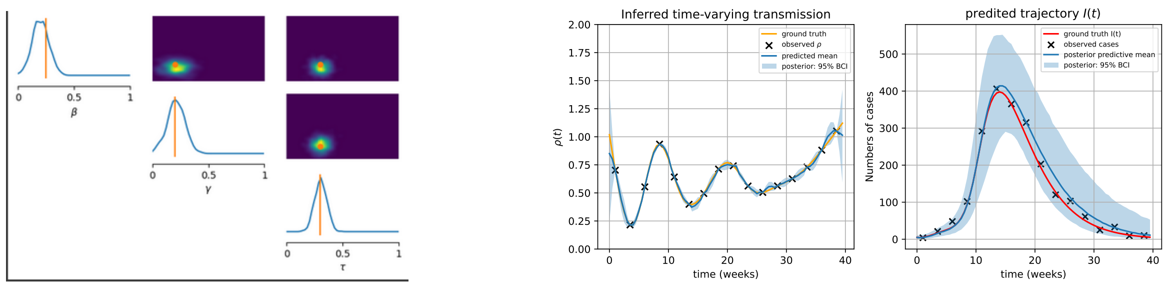

Figure 5.13: Inferred time-varying and time-invariant parameters. Uniform priors were imposed on the constant-in-time parameters. Utilizing 16 confirmed cases, the model accurately reconstructs the time-varying transmission rate, $\rho(t)$, while also recovering the true values for the recovery rate, γ, and the inverse of the average latent period, τ, illustrated by the marginal posterior (kde) analysis plot on the left. The resulting posterior predictive closely aligns with the observed data. The Transformer-based diffusion model effectively handled the SEIR inference task, capturing both the time-invariant parameters (β, γ, τ) and the infinite-dimensional parameter, $\rho(t)$.

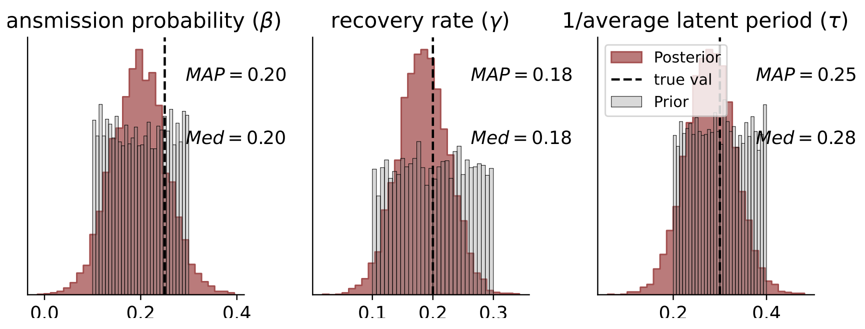

Figure 5.14: Marginal posteriors of time-invariant parameters with priors (gray) and true values (dashed).

6. Discussion and conclusion

In this study, we have introduced new applications of generative models for Bayesian inference in epidemiology, focusing on both spatial models with latent Gaussian variables (GPs) and ODE-based disease models. Specifically, we apply a probabilistic diffusion model built on top of a transformer architecture, as well as a flow-based model implemented as invertible networks for posterior estimation. These approaches overcome several challenges associated with traditional MCMC methods, particularly in spatial modelling. Unlike MCMC, which requires costly computations for each query, generative models can learn from experience and generalize to new observations, significantly reducing computation times.

The transformer-based diffusion model exhibits strong performance and flexibility, especially in managing inference over correlated structures. It allows for querying specific conditionals based on observed data from different locations, offering a new perspective to spatial inference that has typically relied on MCMC methods. Furthermore, this model can accommodate nonparametric ODE-based models, enabling inference with function-valued parameters, expanding its applicability. Similarly, the flow-based model scales efficiently in spatial contexts, enables rapid inference, and amortizes the posterior of compartmental model parameters. This architecture is versatile, applicable to ODE-based time series models, and supports multivariate time series, making it a powerful tool for epidemiological inference.

Limitations The transformer-based diffusion model inherits the challenges associated with both transformers and diffusion models. While generating samples is faster than MCMC-based methods, it remains slower compared to the flow-based model due to the need to solve the reverse SDE. Additionally, transformer evaluations scale quadratically with the number of input tokens, creating significant memory and computational demands during training (Gloeckler et al., 2024a). On the other hand, the flow-based model, though faster for sampling, is less flexible for spatial inference tasks. It tends to produce non-smooth predictions and operates on a case-by-case basis, focusing on one specific set of observations rather than generalizing across spatial domains.

Future work includes leveraging PriorCVAE priors, which are kernel-agnostic, as a surrogate for GP priors to generalize inference across various spatial settings using both approaches (**Sec.** 4.1 , 4.2). This could significantly boost computational efficiency since evaluating the prior would only require sampling from a low-dimensional distribution, $z_d \sim \mathcal{N}(0, I_d)$, followed by a deterministic transformation (see Semenova et al. (2023)). Additionally, improving the conditional invertible architecture for spatial inference could enhance smoothness and allow for amortize inference. Potential solutions include employing random masking in the conditioning module during training to allow generalization, replacing affine transformations with spline-based transformations in the coupling layers, and designing transformation chains that prevent modifications to the observed variables during training and inference.

Conclusion We leverage the capabilities of invertible networks and a state-of-the-art architecture that combines transformers and diffusion models (Simformer) for Bayesian inference in epidemiology within the simulation-based inference framework. Although both methods aim to address the same underlying task, they do so through different strategies, each offering unique benefits suited to various aspects of the problem. This study presented both approaches to highlight their respective contributions to overcoming limitations associated with the primarily used inference methods that rely on MCMC. Key features include speed, amortization properties, efficiency, flexibility, and the ability to perform inference in infinite-dimensional parameter spaces as well as to infer specified conditionals in spatial settings. Immediate potential applications encompass interpolation and extrapolation tasks in epidemiology.

Code accessibility

The codes for reproducing all the results and figures are available at:
https://colab.research.google.com/drive/19hff5YH-5k-3F3dt-1xjQs9mhS0RqOcU (GPs experiment)
https://colab.research.google.com/drive/1rXSdHtVGEbyT7HCPWOqMMaZou1Nk (SIR experiments) .
We used JAX (Bradbury et al., 2018) and PyTorch as the primary frameworks . Some helper functions were adapted from the Simformer GitHub repository (Gloeckler et al., 2024b). The code is optimized for parallel computing on multiple cores, and we strongly recommend running the transformer-based diffusion model on a CUDA-capable machine for efficient performance.

Acknowledgements

I would like to express my sincere gratitude to AIMS and its funders for their invaluable support in facilitating this work. I am also thankful to Google DeepMind for their technical assistance, which has been instrumental to the project's success. Additionally, I extend my heartfelt thanks to my supervisors, Dr. Elizaveta and Dr. Ayush, for their insightful guidance and mentorship throughout this journey.

References

Alsing, J., Wandelt, B., and Feeney, S. 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.

Anselin, L., Syabri, I., and Kho, Y. Geoda: an introduction to spatial data analysis. In *Handbook of applied spatial analysis: Software tools, methods and applications*, pages 73–89. Springer, 2009.

Beaumont, M. A. Approximate bayesian computation in evolution and ecology. *Annual review of ecology, evolution, and systematics*, 41(1):379–406, 2010.

Beaumont, M. A., Zhang, W., and Balding, D. J. Approximate bayesian computation in population genetics. *Genetics*, 162(4):2025–2035, 2002.

Bishop, C. M. Mixture density networks. 1994.

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

Blum, M. G. and François, O. Non-linear regression models for approximate bayesian computation. *Statistics and computing*, 20:63–73, 2010.

Bogachev, V. I. and Ruas, M. A. S. *Measure theory*, volume 1. Springer, 2007.

Bouman, J. A., Hauser, A., Grimm, S. L., Wohlfender, M., Bhatt, S., Semenova, E., Gelman, A., Althaus, C. L., and Riou, J. Bayesian workflow for time-varying transmission in stratified compartmental infectious disease transmission models. *PLoS computational biology*, 20(4):e1011575, 2024.

Box, G. E. and Tiao, G. C. *Bayesian inference in statistical analysis*. John Wiley & Sons, 2011.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., et al. Jax: composable transformations of python+ numpy programs. 2018.

Brauer, F. Compartmental models in epidemiology. *Mathematical epidemiology*, pages 19–79, 2008.

Brehmer, J., Mishra-Sharma, S., Hermans, J., Louppe, G., and Cranmer, K. Mining for dark matter substructure: Inferring subhalo population properties from strong lenses with machine learning. *The Astrophysical Journal*, 886(1):49, 2019.

Chen, Y.-C., Lu, P.-E., Chang, C.-S., and Liu, T.-H. A time-dependent sir model for covid-19 with undetectable infected persons. *Ieee transactions on network science and engineering*, 7(4):3279–3294, 2020.

Clarté, G., Robert, C. P., Ryder, R. J., and Stoehr, J. Componentwise approximate bayesian computation via gibbs-like steps. *Biometrika*, 108(3):591–607, 2021.

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

Dellaportas, P. and Roberts, G. O. An introduction to mcmc. In *Spatial statistics and computational methods*, pages 1–41. Springer, 2003.

Diggle, P. J. and Gratton, R. J. Monte carlo methods of inference for implicit statistical models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 46(2):193–212, 1984.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Germain, M., Gregor, K., Murray, I., and Larochelle, H. Made: Masked autoencoder for distribution estimation. In *International conference on machine learning*, pages 881–889. PMLR, 2015.

Gloeckler, M., Deistler, M., Weilbach, C., Wood, F., and Macke, J. H. All-in-one simulation-based inference. *arXiv preprint arXiv:2404.09636*, 2024a.

Gloeckler, M., Deistler, M., Weilbach, C., Wood, F., and Macke, J. H. All-in-one simulation-based inference, 2024b.

Gonçalves, P. J., Lueckmann, J.-M., Deistler, M., Nonnenmacher, M., Öcal, K., Bassetto, G., Chintaluri, C., Podlaski, W. F., Haddad, S. A., Vogels, T. P., et al. Training deep neural density estimators to identify mechanistic models of neural dynamics. *Elife*, 9:e56261, 2020.

Greenberg, D., Nonnenmacher, M., and Macke, J. Automatic posterior transformation for likelihood-free inference. In *International Conference on Machine Learning*, pages 2404–2414. PMLR, 2019.

Hammersley, J. *Monte carlo methods*. Springer Science & Business Media, 2013.

Haussmann, U. G. and Pardoux, E. Time reversal of diffusions. *The Annals of Probability*, pages 1188–1205, 1986.

Huang, C.-W., Dinh, L., and Courville, A. Augmented normalizing flows: Bridging the gap between generative flows and latent variable models. *arXiv preprint arXiv:2002.07101*, 2020.

Hyvärinen, A. and Dayan, P. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

Kermack, W. O. and McKendrick, A. G. A contribution to the mathematical theory of epidemics. *Proceedings of the royal society of london. Series A, Containing papers of a mathematical and physical character*, 115(772):700–721, 1927.

Kingma, D. P. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Lotka, A. J. Fluctuations in the abundance of a species considered mathematically. *Nature*, 119(2983): 12–12, 1927.

Lueckmann, J.-M., Boelts, J., Greenberg, D., Goncalves, P., and Macke, J. Benchmarking simulation-based inference. In *International conference on artificial intelligence and statistics*, pages 343–351. PMLR, 2021.

Marjoram, P., Molitor, J., Plagnol, V., and Tavaré, S. Markov chain monte carlo without likelihoods. *Proceedings of the National Academy of Sciences*, 100(26):15324–15328, 2003.

Martínez, M. G., Pérez-Castro, E., Reyes-Carreto, R., and Acosta-Pech, R. Spatial modeling in epidemiology. In *Recent Advances in Medical Statistics*. IntechOpen, 2022.

Minka, T. P. Expectation propagation for approximate bayesian inference. *arXiv preprint arXiv:1301.2294*, 2013.

Papamakarios, G. and Murray, I. Fast ε-free inference of simulation models with bayesian conditional density estimation. *Advances in neural information processing systems*, 29, 2016.

Papamakarios, G., Pavlakou, T., and Murray, I. Masked autoregressive flow for density estimation. *Advances in neural information processing systems*, 30, 2017.

Papamakarios, G., Sterratt, D., and Murray, I. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In *The 22nd international conference on artificial intelligence and statistics*, pages 837–848. PMLR, 2019.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

Phillips, R., Kondev, J., Theriot, J., and Garcia, H. *Physical biology of the cell*. Garland Science, 2012.

Radev, S. T., Graw, F., Chen, S., Mutters, N. T., Eichel, V. M., Bärnighausen, T., and Köthe, U. Outbreakflow: Model-based bayesian inference of disease outbreak dynamics with invertible neural networks and its application to the covid-19 pandemics in germany. *PLoS computational biology*, 17 (10):e1009472, 2021.

Rasmussen, C. E. and Nickisch, H. Gaussian processes for machine learning (gpml) toolbox. *The Journal of Machine Learning Research*, 11:3011–3015, 2010.

Rudin, W. et al. *Principles of mathematical analysis*, volume 3. McGraw-hill New York, 1964.

Semenova, E., Xu, Y., Howes, A., Rashid, T., Bhatt, S., Mishra, S., and Flaxman, S. Priorvae: encoding spatial priors with variational autoencoders for small-area estimation. *Journal of the Royal Society Interface*, 19(191):20220094, 2022.

Semenova, E., Verma, P., Cairney-Leeming, M., Solin, A., Bhatt, S., and Flaxman, S. Priorcvae: scalable mcmc parameter inference with bayesian deep generative modelling. *arXiv preprint arXiv:2304.04307*, 2023.

Shun, Z. and McCullagh, P. Laplace approximation of high dimensional integrals. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 57(4):749–760, 1995.

Sohn, K., Lee, H., and Yan, X. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28, 2015.

Stephenson, W. T., Ghosh, S., Nguyen, T. D., Yurochkin, M., Deshpande, S. K., and Broderick, T. Measuring the robustness of gaussian processes to kernel choice. *arXiv preprint arXiv:2106.06510*, 2021.

Talts, S., Betancourt, M., Simpson, D., Vehtari, A., and Gelman, A. Validating bayesian inference algorithms with simulation-based calibration. *arXiv preprint arXiv:1804.06788*, 2018.

Tavaré, S. On the history of abc. In *Handbook of Approximate Bayesian Computation*, pages 55–69. Chapman and Hall/CRC, 2018.

Van Erven, T. and Harremos, P. Rényi divergence and kullback-leibler divergence. *IEEE Transactions on Information Theory*, 60(7):3797–3820, 2014.

Vaswani, A. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Vincent, P. A connection between score matching and denoising autoencoders. *Neural computation*, 23 (7):1661–1674, 2011.

Wan, N., Li, D., and Hovakimyan, N. F-divergence variational inference. *Advances in neural information processing systems*, 33:17370–17379, 2020.

Weilbach, C. D., Harvey, W., and Wood, F. Graphically structured diffusion models. In *International Conference on Machine Learning*, pages 36887–36909. PMLR, 2023.

Williams, C. K. and Rasmussen, C. E. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

Yadan, O. Hydra-a framework for elegantly configuring complex applications, 2019.

Yang, R., Wang, Z., Jiang, B., and Li, S. The convergence of variance exploding diffusion models under the manifold hypothesis.

Appendix A. Appendices

A.1 Affine Coupling Layer with contextual information

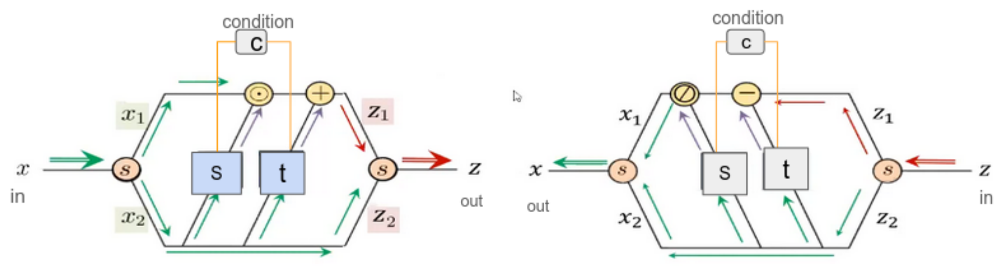

Figure A.1: Forward flow (left) and inverse flow (right) of an affine coupling layer.

- Each coupling layer splits its input $\mathbf{x} \in \mathbb{R}^D$ into two halves $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^{D/2}$.
- Each half is subjected to an affine transformation, producing outputs $\mathbf{z}_1, \mathbf{z}_2 \in \mathbb{R}^{D/2}$.
- Affine coefficients are computed by standard fully connected networks $\mathbf{s} \in \mathbb{R}_+^{D/2}$ and $\mathbf{t} \in \mathbb{R}^{D/2}$ from the other half's data.

$$\textit{Forward computation}: z = f(x, c) \iff z_1 = x_1 \odot s(x_2, c) + t(x_2, c), \quad z_2 = x_2$$
$$\textit{Inverse computation}: x = f^{-1}(z, c) \iff x_1 = (z_1 - t(z_2, c)) \oslash s(z_2, c), \quad x_2 = z_2$$

$c = g_\eta(y)$ is potentially the resultant feature vector of a condition y obtained via a conditioning net g_η.

A.2 Approximate Bayesian Computation

Approximate Bayesian Computation (ABC) (Beaumont, 2010) is one of the traditional methods used to address the problem of intractable likelihood in Bayesian inference. The first proposed method in this context, known as *Rejection ABC* (Tavaré, 2018), was initially related to a genetic problem. The simplified algorithm is as follows:

Given the observation y_{obs}, repeat the following until N points have been accepted:

1. Sample $\theta \sim p(\theta)$
2. Use that θ to simulate y_{sim} using the simulator
3. If $y_{sim} = y_{obs}$,accept θ as a sample point from posterior, otherwise discard that θ

The posterior distribution from which these points are sampled provides the probability distribution of the parameter value that generated the observations. A significant challenge arises in more complex scenarios; for example, when y is continuous, the probability of y_{sim} being exactly equal to the observed y_{obs} is zero, i.e., $p(y_{sim} = y_{obs}) = 0$. To address this issue, the algorithm has been modified to accept not only exact values but also "*close-enough*" values by introducing a threshold ϵ and a distance metric ρ (e.g., $\| \,.\, \|$). Given an observation y_{obs}, repeat the following until N points have been accepted:

1. Sample $\theta \sim p(\theta)$
2. $y_{sim} \sim p(\cdot|\theta)$
3. If $\rho(S(y_{sim}), S(y_{obs})) < \epsilon$, accept θ as a sample point from posterior, otherwise discard that θ.

where S is a low-dimensional summary statistic. This algorithm does not sample from the exact posterior but rather from an approximate posterior. As $\epsilon \to 0$, the samples are drawn from the true posterior. However, there is a tradeoff: as ϵ decreases, the number of rejected samples increases while the number of accepted samples decreases. Thus, to obtain samples very close to the true posterior, we need to lower ϵ, which may result in a longer wait for a sufficient number of accepted samples. Typically, we choose a low enough ϵ to maintain a reasonable acceptance rate, even if it means sacrificing some accuracy in the posterior estimation.

Rejection-ABC can be seen as a method of performing rejection sampling while addressing the challenge of not knowing the likelihood $p(y \mid \theta)$. If we adopt this approach, we might also consider using $MCMC$ or SMC methods, which offer benefits over traditional rejection sampling, such as increased acceptance rates. The algorithms that facilitate this are *MCMC-ABC* (Marjoram et al., 2003) and *SMC-ABC* (Clarté et al., 2021). In any of these methods—*Rejection-ABC, MCMC-ABC*, or *SMC-ABC*—we input θ into a simulator to generate y_{sim} and then seek the values that yield results similar to the observed data y_{obs}. This process implies a relationship between y and θ; specifically, for any given y, there exists a distribution over θ that could plausibly produce that y. *Regression Adjustment* by Beaumont et al. (2002) involves hypothesizing about the nature of this relationship. The simplest structure, linear regression, posits that changes in y affect the mean of the posterior distribution in a linear manner while maintaining constant variance (homoscedasticity). This approach has been enhanced to *weighted linear regression* (Beaumont et al., 2002) and further extended to learn nonlinear structures with heteroscedastic variance using neural networks Blum and François (2010). A significant advantage of this approach is that it allows for higher threshold values, thereby improving the capacity to handle larger dimensions compared to previous ABC algorithms.However, getting the model structure wrong will result on worst approximation.

The *ABC* Approximate Bayesian Computation approaches face the curse of dimensionality: in the worst-case scenario, the number of required simulations increases exponentially with the data's dimensionality. As a result, these methods rely on low-dimensional summary statistics, $S(y)$, and the quality of the inference is closely tied to how well these summaries retain information about the parameters, θ. Recently, more flexible neural network-based solutions have emerged to address these shortcomings.

A.3 Score Matching

Suppose we have our continuous probability distribution where we use $p(x)$ to represent the probability distribution. The score function is defined as the gradient of $\nabla_x \log p(x)$ with respect to the random variable x. It is the vector field that gives the direction where the density function grows most quickly. Mathematically, this score function preserves all the information about the density $p(x)$. When using scores to represent probability distributions, we develop our score-based generative model, which allows for significant flexibility. This flexibility arises from the fact that the score function does not need to be normalized. As a result, we can employ a variety of neural network architectures to represent this score function. Additionally, we can learn these score function models from data using principle statistical approaches:

$$\nabla_x \log p_\theta(x) = \nabla_x \frac{e^{f_\theta(x)}}{Z_\theta} = \nabla_x f_\theta(x) = s_\theta(x)$$

We use $s_\theta(x)$ to denote such a deep neural network for the score function called score model. Mathematically, given $\{x_1, x_2, ..., x_N\} \overset{i.i.d}{\sim} p_{data}(x)$, we approximate the ground truth score function $\nabla_x \log p_{data}(x) \approx s_\theta(x)$ by minimizing the *Fisher divergence objective* $\frac{1}{2}\mathbb{E}_{p_{data}(x)}[\| \nabla_x \log p_{data}(x) - s_\theta(x) \|_2^2]$. However, the *Fisher divergence objective* cannot be computed directly because the true value of the data function score is unknown. The method known as *score matching* (Hyvärinen and Dayan, 2005) tackles this challenge by applying integration by parts from Gauss's theorem, which allows us to transform the *Fisher divergence* into an equivalent objective:

$$\mathbb{E}_{p_{data}(x)}\Big[\frac{1}{2} \| s_\theta(x) \|_2^2 + \text{trace}(\nabla_x s_\theta(x))\Big] + \text{const} \sim \frac{1}{N}\sum_{i=1}^{N}\Big[\frac{1}{2} \| s_\theta(x_i) \|_2^2 + \text{trace}(\nabla_x s_\theta(x_i))\Big] \quad \text{(A.3.1)}$$

The score matching objective is not scalable to compute, especially when employing deep neural networks for high-dimensional data points. Denoising score matching (Vincent, 2011) offers a solution to this computational challenge. It involves adding extra noise to the data points through a perturbation kernel q_σ (usually Gaussian), which facilitates the calculation of the Jacobian term.

$$p_{data}(x) \longrightarrow q_\sigma(\tilde{x} \mid x) \longrightarrow q_\sigma(\tilde{x})$$

Then, for very small σ (standard deviation of the kernel), we can approximately view the score function of the noisy density as equivalent to the score function of the noise-free density $\nabla_{\tilde{x}} \log q_\sigma(\tilde{x}) \approx \nabla_x p_{data}(x)$. Therefore, the objective becomes

$$\frac{1}{2}\mathbb{E}_{q_\sigma(\tilde{x})}[\| \nabla_{\tilde{x}} \log q_\sigma(\tilde{x}) - s_\theta(\tilde{x}) \|_2^2] = \frac{1}{2}\mathbb{E}_{p_{data}(x)}\Big[\mathbb{E}_{q_\sigma(\tilde{x}|x)}[\| \nabla_{\tilde{x}} \log q_\sigma(\tilde{x} \mid x) - s_\theta(\tilde{x}) \|_2^2]\Big] \quad \text{(A.3.2)}$$

In this new form, all we need to compute is the gradient of the perturbation kernel $\nabla_{\tilde{x}} \log q_\sigma(\tilde{x} \mid x)$, but because we designed it by hand, it is usually fully tractable, so computing this gradient is very efficient.

A.4 Auto-encoding Generative Models

An auto-encoding model is a type of neural network designed to learn efficient representations of data, typically for the purpose of dimensionality reduction or data generation. It consists of two main components: an encoder and a decoder. The encoder compresses the input data into a lower-dimensional latent space, capturing essential features, while the decoder reconstructs the original data from this compressed representation. By minimizing the difference between the input and the reconstructed output during training, the model learns to generate new data samples that resemble the original dataset.

Variational Autoencoders (VAE) introduced by Kingma (2013), combines the principles of autoencoders and variational inference to create autoencoders with a more "regularized" latent space. Unlike traditional autoencoders that encode input data into a deterministic latent space, VAEs encode data into a probabilistic latent space, where each input is represented by a distribution (typically a Gaussian) rather than a fixed point. This probabilistic nature allows VAEs to generate diverse and realistic data samples. The objective function of a VAE is derived from maximizing the Evidence Lower Bound (ELBO) on the log likelihood of the observed data. The ELBO comprises two main components:

- *Reconstruction Loss:* Measures how accurately the decoder can reconstruct the input data from the latent variables. Typically,expressed as the negative log-likelihood of the data given the latent variables.
- *KL Divergence:* Measures how closely the approximate posterior distribution $q(z \mid y)$ matches the true prior distribution $p(z)$. It acts as a regularizer to ensure that the learned latent space has desirable properties.

The ELBO is given by:

$$\mathcal{L}(\theta, \phi; y) = \mathbb{E}_{q_\phi(z|y)}[\log p_\theta(y|z)] - \mathsf{KL}(q_\phi(z|y) \| p(z)) \tag{A.4.1}$$

θ and ϕ are the parameters of the decoder and encoder networks, respectively, y is the observed data, and z is the latent variable. The encoder maps input data to a latent space, producing parameters for the approximate posterior distribution $q(z|y)$, typically modeled as a Gaussian characterized by its mean $\mu(y)$ and variance $\sigma(y)$. The decoder then reconstructs the data from these sampled latent variables. To allow for gradient backpropagation through the stochastic sampling process, the reparameterization trick is employed, expressing z as a deterministic function of the input x and some noise $\epsilon \sim \mathcal{N}(0, I)$: $z = \mu(y) + \sigma(y) \cdot \epsilon$. The latent space is structured using a standard normal prior distribution, $p(z) = \mathcal{N}(0, I_d)$ As per Kingma (2013).

Conditional VAEs (CVAEs) as proposed by Sohn et al. (2015), build upon VAEs by incorporating additional information c (e.g., a class label) into the generative process,. This modification enables CVAEs to handle a range of input-output mappings, as the optimization objective now considers both the data and the conditioning input. Consequently, we gain the ability to control the class from which we desire to generate samples, while also allowing the CVAE to learn distinct encoding and decoding processes for different classes of input. The optimization objective becomes:

$$\mathcal{L}(\theta, \phi; y) = \mathbb{E}_{q_\phi(z|y,c)}[\log p_\theta(y|z, c)] - \mathsf{KL}(q_\phi(z|y, c) \| p(z \mid c)) \tag{A.4.2}$$

A.5 PriorCVAE

To tackle the scalability challenges of Markov Chain Monte Carlo (MCMC) inference when using Gaussian Process (GP) priors, Semenova et al. (2022) introduced PriorVAE, a deep generative model-based solution. PriorVAE utilizes the Variational Autoencoder (VAE) framework to encode GP priors, offering a scalable and efficient representation. Once trained, the generative models act as surrogates for the original GP priors, enabling fast and flexible Bayesian inference on new data. This method was primarily designed to address the small area estimation (SAE) problem in spatial statistics. The workflow is as follows:

1. Define the spatial setting: establish the spatial structure of interest, defined as the set $\{x_1, \dots, x_n\}$, such as a collection of administrative units.

2. Learn uncorrelated representations of spatial GP priors: sample evaluations from a Gaussian process (GP) prior, denoted as $f \sim \mathcal{GP}(.)$, based on the defined spatial structure. Train the Variational Autoencoder (VAE) to approximate these evaluations as $f_{GP} = (f(x_1), \dots, f(x_n))$.

3. Perform inference: utilize the learned decoder $p_\theta(. \mid .)$ in the model as a surrogate of the original GP priors during MCMC inference on new data. Here, $f_{GP} \approx f_{PriorVAE} = p_\theta(. \mid z_d)$, where $z_d \sim \mathcal{N}(0, I_d)$.

While this approach has demonstrated efficient inference capabilities, it has the drawback of losing information about the hyperparameters of the original models. The encoding network, $q_\phi(. \mid f_{GP})$, does not distinguish between samples generated with different hyperparameter values. As a result, inference over hyperparameters becomes impossible, and the learned priors become indistinguishable. To overcome this limitation, Semenova et al. (2023) extended the PriorVAE method to PriorCVAE by conditioning the VAE on stochastic process hyperparameters. The core idea is to jointly encode hyperparameters alongside GP realizations and estimate them during inference.

The PriorCVAE method offers several advantages over PriorVAE. It allows for clear prior identification, enabling downstream MCMC tasks to explicitly estimate hyperparameters. Additionally, it facilitates the model's ability to draw priors for hyperparameter values that were not included in the original training set, resulting in a more robust class of approximated Gaussian Processes.

Furthermore, PriorCVAE is not limited to a specific model. The authors demonstrate this versatility by encoding the solutions of an Ordinary Differential Equation (ODE). This flexibility allows PriorCVAE to be applicable across various models, rather than being restricted to a single use case. In summary, the PriorCVAE approach not only retains the benefits of the PriorVAE method but also introduces unique capabilities for prior identification, hyperparameter estimation, and model-agnostic applicability.

A.6 Additional Findings

	Running Time (s)		Sampling Time (s)	
	cINN	Simformer	cINN	Simformer
One-Dimensional GP	140	2677	10	69
Two-Dimensional GP	200	3256	15	140

Table A.1: Comparison of training and inference times for cINN and Simformer models. Both models were trained on 20,000 realizations of Gaussian Processes (GPs), followed by generating 10,000 posterior samples to evaluate the models' performance.

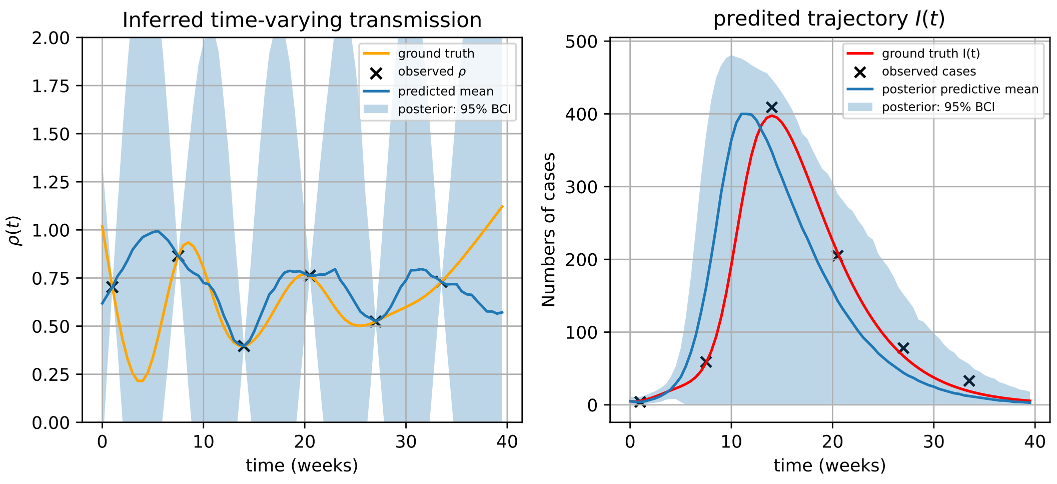

Figure A.2: Inferred time-varying transmission $\rho(t)$ (left) and the posterior predictive plot (right).Based on only five observations, the model poorly infers the transmission pattern over time, leading to significant uncertainty in the posterior predictions.