

Spatial Bayesian Neural Networks

Andrew Zammit-Mangion^{a,*}, Michael D. Kaminski^a, Ba-Hien Tran^b, Maurizio Filippone^b, Noel Cressie^a

^a*School of Mathematics and Applied Statistics, University of Wollongong, Australia*

^b*Data Science Department, EURECOM, Sophia Antipolis, France*

Abstract

Statistical models for spatial processes play a central role in statistical analyses of spatial data. Yet, it is the simple, interpretable, and well understood models that are routinely employed even though, as is revealed through prior and posterior predictive checks, these can poorly characterise the spatial heterogeneity in the underlying process of interest. Here, we propose a new, flexible class of spatial-process models, which we refer to as spatial Bayesian neural networks (SBNNs). An SBNN leverages the representational capacity of a Bayesian neural network; it is tailored to a spatial setting by incorporating a spatial “embedding layer” into the network and, possibly, spatially-varying network parameters. An SBNN is calibrated by matching its finite-dimensional distribution at locations on a fine gridding of space to that of a target process of interest. That process could be easy to simulate from or we have many realisations from it. We propose several variants of SBNNs, most of which are able to match the finite-dimensional distribution of the target process at the selected grid better than conventional BNNs of similar complexity. We also show that a single SBNN can be used to represent a variety of spatial processes often used in practice, such as Gaussian processes and lognormal processes. We briefly discuss the tools that could be used to make inference with SBNNs, and we conclude with a discussion of their advantages and limitations.

Keywords: Gaussian process; Hamiltonian Monte Carlo; Lognormal process; Non-stationarity; Wasserstein distance

1. Introduction

At the core of most statistical analyses of spatial data is a spatial process model. The spatial statistician has a wide array of models to choose from, ranging from the ubiquitous Gaussian-process models (e.g., Rasmussen and Williams, 2006) to the trans-Gaussian class of spatial models (e.g., De Oliveira et al., 1997), all the way to the more complicated models for spatial extremes (e.g., Davison and Huser, 2015). Each model class is itself rich, which leads to several questions: What class of covariance functions should be used? Should the model account for anisotropy or non-stationarity? Should the model be non-Gaussian? Will the model lead to computationally efficient inferences? There is a long list of exploratory checks and modelling decisions to be made when undertaking a spatial statistical analysis (Cressie, 1993), and these need to be done every time the spatial statistician is presented with new data. The typical workflow requires one to be cognisant of the large array of models available, paired with computational tools that are themselves varied and complex.

*Corresponding author

Email addresses: azm@uow.edu.au (Andrew Zammit-Mangion), mdk201@uowmail.edu.au (Michael D. Kaminski), ba-hien.tran@eurecom.fr (Ba-Hien Tran), maurizio.filippone@eurecom.fr (Maurizio Filippone), ncressie@uow.edu.au (Noel Cressie)

In this paper we outline a new paradigm for spatial statistical analysis that hinges on the use of a single, highly adaptive, class of spatial process models, irrespective of the data or the application. We refer to a member of this model class as a spatial Bayesian neural network (SBNN), which we *calibrate* by matching its distribution at locations on a fine gridding of space to that of a spatial process of interest. An SBNN is parameterised using weights and biases, but the parameterisation is high-dimensional and does not change with the context in which it is used. Consequently, the same calibration techniques and algorithms can be used irrespective of the application and the spatial process of interest. The primary purpose of modelling with an SBNN is to relieve the spatial statistician from having to make difficult modelling and computing decisions. At the core of an SBNN is a Bayesian neural network (BNN, [Neal, 1996](#)), that is, a neural network where the network parameters (i.e., the weights and the biases) are endowed with a prior distribution.

Bayesian neural networks are not new to spatial or spatio-temporal statistics; see, for example [McDermott and Wikle \(2019\)](#) for their use in the context of spatio-temporal forecasting; [Payares-Garcia et al. \(2023\)](#) for their use in neurodegenerative-disease classification from magnetic resonance images; and [Kirkwood et al. \(2022\)](#) for their use in geochemical mapping. However, their use to date involves models constructed through rather simple prior distributions over the weights and parameters: The priors are typically user-specified, fixed, or parameter invariant. Such a prior distribution, although straightforward to define, will likely lead to a prior spatial process model that is degenerate or, at the very least, highly uncharacteristic of the process of interest; see, for example, the related discussion of [Duvenaud et al. \(2014\)](#) in the context of deep neural networks and that of [Dunlop et al. \(2018\)](#) in the context of deep Gaussian processes. This degeneracy stems from the fact that neural networks are highly nonlinear, and that the prior distribution on the weights and biases has a complicated, and unpredictable, effect on the properties of the spatial process. While a poor choice of spatial process model may be inconsequential when large quantities of data are available, it is likely to be problematic when data are scarce or reasonably uninformative of the model's parameters.

The core idea that we leverage in this work is the approach of [Tran et al. \(2022\)](#), which we summarise as follows: If one has a sufficiently large number of realisations from a target process of interest, then one can calibrate a BNN such that its finite-dimensional distributions closely follow those of the underlying target process. Once this calibration is done, inferential methods for finding the posterior distribution of the parameters in the BNN can be used, after new data are observed; these methods generally involve stochastic gradient Hamiltonian Monte Carlo (SGHMC, [Chen et al., 2014](#)) or variational Bayes (e.g., [Graves, 2011](#); [Zammit-Mangion et al., 2022](#)). This paradigm is attractive: from a modelling perspective, one does not need to look for an appropriate class of spatial process models every time a new analysis is needed, and the same inferential tool can be used irrespective of the data being considered. From a deep-learning perspective, calibration avoids the pathologies that one risks when applying BNNs with fixed priors to spatial or spatio-temporal analyses. The method of [Tran et al. \(2022\)](#) is novel in its approach of enabling computationally- and memory-efficient calibration via a Monte Carlo approach. However, the BNNs they consider are not directly applicable to spatial problems: A novelty of our SBNNs is that they incorporate a spatial embedding layer and spatially-varying network parameters, which we show help capture spatial covariances and non-stationary behaviour of the underlying target spatial process. Specifically, we show that several such SBNNs are able to match a selected high- and finite-dimensional distribution of the target spatial process better than conventional BNNs of similar complexity.

In Section 2 we motivate and construct SBNNs, while in Section 3 we detail an approach for calibrating SBNNs that follows closely that of [Tran et al. \(2022\)](#). In Section 4, we show that our SBNNs can be used as high-quality surrogate models for both stationary and highly non-stationary Gaussian processes, as well as lognormal spatial processes. In Section 5 we outline how SBNNs can be used in practice; namely, as stochastic generators, and for making inference.

Finally, in Section 6 we conclude with a discussion of the benefits and limitations of SBNNs.

2. Methodology

2.1. Bayesian neural networks for spatial data

A BNN (Neal, 1996, pp. 10–19) is a composition of nonlinear random functions. Each function comprises a so-called ‘layer’ of the network. In the context of spatial data analysis, a BNN is used to model a spatial process $Y(\cdot)$ over a spatial domain $D \subset \mathbb{R}^d$, where the spatial dimension d is small; typically, $d \in \{1, 2, 3\}$. We define a BNN as follows:

$$Y(\mathbf{s}) \equiv f(\mathbf{s}; \boldsymbol{\theta}) = f_L(\mathbf{f}_{L-1}(\cdots \mathbf{f}_1(\mathbf{f}_0(\mathbf{s}; \theta_0); \boldsymbol{\theta}_1) \cdots; \boldsymbol{\theta}_{L-1}); \boldsymbol{\theta}_L), \quad \mathbf{s} \in D, \quad (1)$$

where, for $l = 0, \dots, L$, $\mathbf{f}_l(\cdot; \boldsymbol{\theta}_l) \in \mathbb{R}^{d_l}$, d_l is the output dimension of the l -th layer, and $d_L = 1$. We parameterise each function $f_l(\cdot; \boldsymbol{\theta}_l)$ by $\boldsymbol{\theta}_l$, for $l = 0, \dots, L$. We will assume that θ_0 is known by design, and we collect the remaining parameters across the whole BNN into $\boldsymbol{\theta} \equiv \{\boldsymbol{\theta}_l : l = 1, \dots, L\}$. For what we refer to as a *vanilla* BNN, we let $\mathbf{f}_0(\mathbf{s}; \theta_0) \equiv \mathbf{s}$, for $\mathbf{s} \in D$ (so that $d_0 = d$), and we let the remaining layers have the form

$$\mathbf{f}_l(\cdot; \boldsymbol{\theta}_l) = \frac{1}{\sqrt{d_{l-1}}} \mathbf{W}_l \boldsymbol{\varphi}_{l-1}(\cdot) + \mathbf{b}_l, \quad l = 1, \dots, L, \quad (2)$$

where $\boldsymbol{\theta}_l \equiv \{\boldsymbol{\theta}_l^w, \boldsymbol{\theta}_l^b\}$; $\boldsymbol{\theta}_l^w \equiv \text{vec}(\mathbf{W}_l)$; $\boldsymbol{\theta}_l^b \equiv \mathbf{b}_l$; $\mathbf{W}_l \in \mathbb{R}^{d_l \times d_{l-1}}$ and $\mathbf{b}_l \in \mathbb{R}^{d_l}$ are random weights and biases associated with the l -th layer, respectively, to which prior distributions will be assigned; and $\text{vec}(\cdot)$ is the vectorisation operator. Further, in (2), $\boldsymbol{\varphi}_{l-1}(\cdot) \in \mathbb{R}^{d_{l-1}}$ are so-called activation functions that are evaluated element-wise over the output of layer $l-1$; popular activation functions include the hyperbolic tangent, the rectified linear unit (ReLU), and the softplus functions (Goodfellow et al., 2016). Finally, the division by $\sqrt{d_{l-1}}$ in (2) is referred to as the neural-tangent-kernel (NTK) parameterisation; see Jacot et al. (2018) for its motivation and further details.

The finite-dimensional distributions of the process $Y(\cdot)$ are fully determined by the prior distribution of the weights and biases that define $\boldsymbol{\theta}$. A natural question is then: What distribution should we have for the weights and biases? There is no straightforward answer to this question, in large part because of the nonlinearities inherent to BNNs, which make the relationship between the prior distribution of $\boldsymbol{\theta}$ and that of the process $Y(\cdot)$ unintuitive. Yet, the choice is important: For example, one might model the weights and biases as independent and equip them with a $\text{Gau}(0, 1)$ distribution, but this choice leads to a seemingly degenerate stochastic process. For illustration, Figure 1 shows sample paths drawn from the process $Y(\cdot)$ over $D \equiv [-4, 4]$ when all weights and biases are simulated independently from a $\text{Gau}(0, 1)$ distribution. Observe that, as the number of layers increases from $L = 1$ (top-left panel) to $L = 8$ (bottom-right panel), sample paths from the process tend to flatten out over $\mathbf{s} \in D$. Clearly, the stochastic process with $L = 8$ is an unreasonable model for applications involving spatial data.

In Section 3 we show how one may *calibrate* these prior distributions (i.e., estimate the hyper-parameters that parameterise the prior means and variances) so that a selected high- and finite-dimensional distribution from $Y(\cdot)$ – for example one at locations on a fine gridding of D – matches closely that of another, user-specified, spatial process. As is shown in Section 4, we had difficulty calibrating the vanilla BNN given by (1) to spatial models that are routinely employed (e.g., the Gaussian process), in the sense that we were unable to obtain a BNN with a finite-dimensional distribution that was close to that of the target spatial model. This was likely because the vanilla BNN in (1) is not tailored for spatial data. Our SBNNs modify (1) in two ways that specifically aim at modelling spatial dependence. These modifications lead to a class of SBNNs that can match target spatial processes more closely than BNNs of similar complexity; these modifications are discussed in the following section.

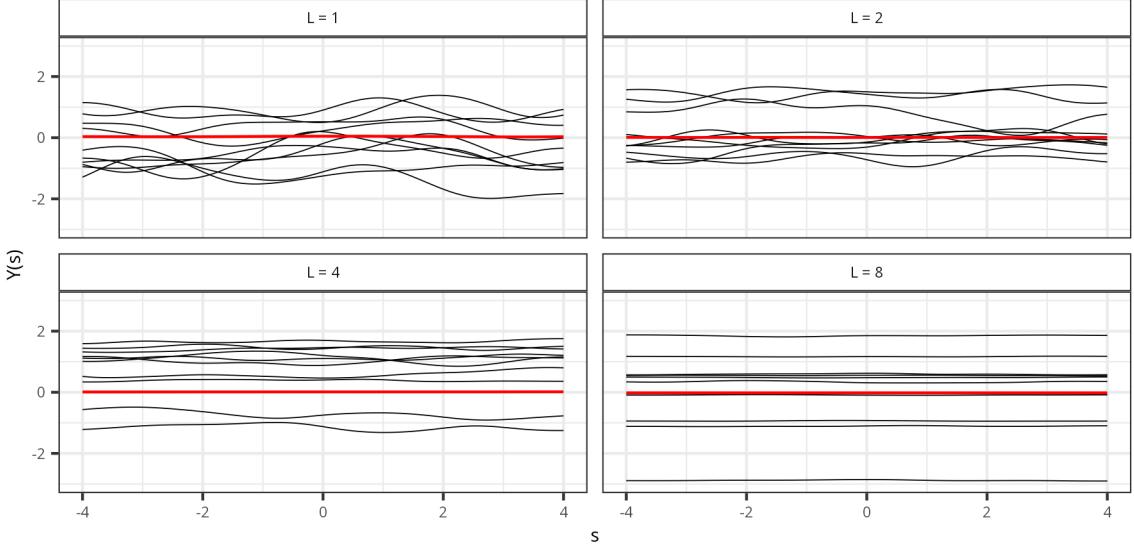


Fig. 1: Sample paths (black) of a vanilla Bayesian neural network of the form (1) and (2) with $\mathbf{f}_0(\mathbf{s}; \theta_0) \equiv \mathbf{s}$, where each hidden layer is of dimension $d_l = 40$, $l = 1, \dots, (L-1)$, the activation functions in $\varphi_{l-1}(\cdot)$, $l = 1, \dots, L$, are $\tanh(\cdot)$ functions, $D \equiv [-4, 4]$, and the weights and biases collected in θ are all drawn independently from a standard normal distribution with zero mean and unit variance. The empirical-process mean computed from 4000 sample paths is also shown (red). (Top-left panel) $L = 1$. (Top-right panel) $L = 2$. (Bottom-left panel) $L = 4$. (Bottom-right panel) $L = 8$.

2.2. Spatial Bayesian neural networks

As shown later in Section 4, we often found difficulty capturing spatial covariances when using vanilla BNNs (i.e., (1) with $\mathbf{f}_0(\mathbf{s}; \theta_0) \equiv \mathbf{s}$, $\mathbf{s} \in D$). This corroborates the findings of Chen et al. (2023), who argue that classical neural networks cannot easily incorporate spatial dependence between the inputs, when employed for spatial prediction. In their paper, they mitigated this issue by using a set of spatial basis functions in the first layer of the network, in a process they called *deepKriging*. We also found that the inclusion of this ‘‘embedding layer’’ greatly improved the ability of SBNNs to express realistic covariances (Section 2.2.1). However, we found that, even with the embedding layer, our SBNNs were prone to not capturing complex non-stationary behaviour. To deal with this, we made the parameters appearing in our SBNNs spatially varying (Section 2.2.2).

2.2.1. The embedding layer in an SBNN

The first modification we make to the classical BNN is to follow Chen et al. (2023) and replace the zeroth layer in (1) with an embedding layer comprising a set of K radial basis functions (RBF, Buhmann, 2003) centred on a regularly-spaced grid in D . Specifically, we define $\mathbf{f}_0(\mathbf{s}; \theta_0) \equiv \boldsymbol{\rho}(\mathbf{s}; \tau) \equiv (\rho_1(\mathbf{s}; \tau), \dots, \rho_K(\mathbf{s}; \tau))'$, where $\rho_k(\cdot; \tau)$, $k = 1, \dots, K$, are K radial basis functions with centroids $\{\boldsymbol{\xi}_k \in D : k = 1, \dots, K\}$ that together make up the spatial embedding layer, and $\theta_0 = \tau > 0$ is a length-scale parameter. Here, we use the Gaussian bell-shaped radial basis functions; specifically, we let

$$\rho_k(\mathbf{s}; \tau) = \exp \left[- \left(\frac{\|\mathbf{s} - \boldsymbol{\xi}_k\|}{\tau} \right)^2 \right], \quad k = 1, \dots, K, \quad (3)$$

where $\|\cdot\|$ is the Euclidean norm. We set τ so that the radial basis functions have a suitable level of overlap, as is often done in low-rank spatial modelling (e.g., Cressie and Johannesson, 2008; Nychka et al., 2018; Zammit-Mangion and Cressie, 2021); see Figure S1 in the Supplementary

Material for an example. Given a spatial location $\mathbf{s} \in \mathbb{R}^d$, the output of the embedding layer is given by $\rho(\mathbf{s}; \tau) \in \mathbb{R}^{d_0}$ with $d_0 = K$. The spatial embedding layer therefore represents a vector $\rho(\mathbf{s}; \tau) \in (0, 1]^K$, indexed by spatial location \mathbf{s} , which encodes the proximity of the given input to each RBF centroid. A value for $\rho_k(\mathbf{s}; \tau)$ close to one indicates proximity of the point \mathbf{s} to the centroid ξ_k and, as \mathbf{s} moves further away from ξ_k , its value decreases rapidly. The weights and biases of an SBNN are drawn from a spatially-invariant distribution, and we therefore refer to the resulting network as an SBNN with spatially-invariant parameters (SBNN-I).

2.2.2. Spatially varying network parameters

In Section 4 we show that the embedding layer is generally important for an SBNN to model covariances, but we also show that to model non-stationarity one needs something more. One natural way to introduce additional flexibility to capture spatially heterogeneous behaviour better, is to make the weights and biases of the SBNN-I spatially varying.

We do this by defining the distribution for each weight and bias as Gaussian with some mean and variance that both vary smoothly over D . As a practical matter, we employ the same basis used in the embedding layer, $\rho(\cdot; \tau)$, to model the smoothly-varying means and variances.

For ease of notation, consider a single, now spatially-varying, weight or bias parameter $\theta(\cdot)$. We model the prior mean of $\theta(\cdot)$ as

$$\mu(\cdot) = \rho_1(\cdot)\alpha_1 + \cdots + \rho_K(\cdot)\alpha_K, \quad (4)$$

where $\alpha_k \in \mathbb{R}$ for $k = 1, \dots, K$, and the prior standard deviation as

$$\sigma(\cdot) = \text{softplus}(\rho_1(\cdot)\beta_1 + \cdots + \rho_K(\cdot)\beta_K), \quad (5)$$

where $\beta_k \in \mathbb{R}$ for $k = 1, \dots, K$ and $\text{softplus} : t \mapsto \ln(1 + e^t)$ enforces positivity. It remains then to establish the spatial covariance of $\theta(\cdot)$

which, for $\eta \sim \text{Gau}(0, 1)$, we define as follows:

$$\theta(\cdot) = \mu(\cdot) + \sigma(\cdot)\eta. \quad (6)$$

Under this model, for any two locations $\mathbf{s} \in D$ and $\mathbf{r} \in D$, $\text{cov}(\theta(\mathbf{s}), \theta(\mathbf{r})) = \sigma(\mathbf{s})\sigma(\mathbf{r})$ and hence $\text{corr}(\theta(\mathbf{s}), \theta(\mathbf{r})) = 1$. This is a rather inflexible prior (spatial) model, but it comes with the advantage of not introducing additional covariance hyper-parameters that would otherwise need to be estimated. All smoothness in $\theta(\cdot)$ for a given weight or bias is induced by that in its $\mu(\cdot)$ and $\sigma(\cdot)$, which are smooth by construction. Since there are many weights and biases (i.e., many $\theta(\cdot)$'s), there are many η 's (one for each weight and bias), which we model as mutually independent. Rather than estimating scalar means and standard deviations when calibrating, we instead estimate the coefficients α_k and β_k , for $k = 1, \dots, K$, for each weight and bias parameter, or groups thereof. We refer to the resulting network as an SBNN with spatially-varying parameters (SBNN-V).

We illustrate the architecture of the SBNN-V in Figure 2. Note that by setting the means and standard deviations of the neural-network parameters to be of the form (4) and (5), we are introducing so-called *skip connections* into the network architecture, which feed the output of the embedding layer $\rho(\cdot; \tau)$ directly into each subsequent layer. This use of skip connections is analogous to the way they are used in the popular architecture *ResNet* for feature re-usability (He et al., 2016). This SBNN-V can also be viewed as a simple *hyper-network*, since the prior means and standard deviations of the weights and biases are themselves the outputs of shallow (one-layer) networks (e.g., Malinin et al., 2020).

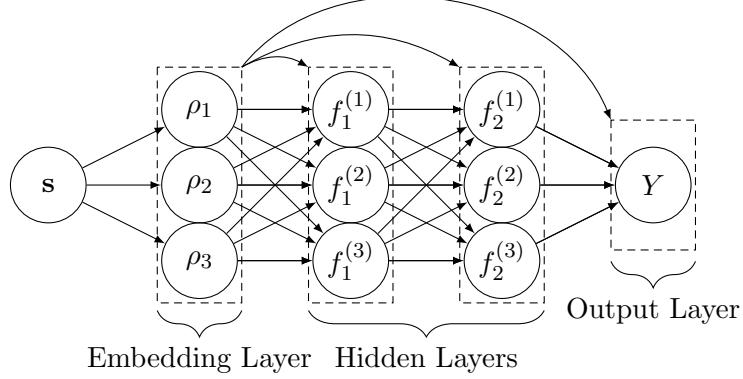


Fig. 2: Schematic of a $L = 3$ -layer SBNN with spatially-varying parameters (SBNN-V). The SBNN-V contains an embedding layer and skip connections that feed the output of the embedding layer into each subsequent layer of the network.

2.2.3. Model specification of (S)BNNs

An SBNN comprises many (potentially several thousand) weights and biases which, as is typical with BNNs, we model as mutually independent. However, a choice must be made on whether to equip all of these independent parameters with different prior distributions, or whether to assume that the parameters are independent but identically distributed within groups. In this work we will consider both options. The former ‘prior-per-parameter’ scheme comes with the advantage that it leads to a highly flexible SBNN, with the downside that many hyper-parameters need to be stored and estimated during calibration. For the latter option we will group the parameters by layer in what we refer to as a ‘prior-per-layer’ scheme (note that other groupings are possible; see, for example MacKay, 1992). The ‘prior-per-layer’ scheme comes with the advantage that the number of hyper-parameters that need to be estimated during calibration is much smaller; an SBNN constructed under this scheme is therefore easier and faster to calibrate. However, we find in Section 4 that while this ‘prior-per-layer’ scheme is sufficiently flexible to model some stochastic processes of interest, there are cases where SBNNs with a ‘prior-per-parameter’ scheme may calibrate better to the target process. In what follows we detail our SBNNs under both schemes, the ‘prior-per-layer’ scheme (“SBNN-IL” and “SBNN-VL”) and the ‘prior-per-parameter’ scheme (“SBNN-IP” and “SBNN-VP”). For completeness we also outline the vanilla BNN variants, whose parameters are spatially invariant by definition (“BNN-IL” and “BNN-IP”).

SBNN-IL: The SBNN-I with a ‘prior-per-layer’ scheme (SBNN-IL) is given by the following hierarchical spatial statistical model:

For $\mathbf{s} \in D$, $l = 1, \dots, L$,

$$\begin{aligned} Y(\mathbf{s}) &\equiv f(\mathbf{s}; \boldsymbol{\theta}) = f_L(\mathbf{f}_{L-1}(\dots \mathbf{f}_1(\mathbf{f}_0(\mathbf{s}; \boldsymbol{\theta}_0); \boldsymbol{\theta}_1) \dots; \boldsymbol{\theta}_{L-1}); \boldsymbol{\theta}_L), \\ \mathbf{f}_l(\cdot; \boldsymbol{\theta}_l) &= \frac{1}{\sqrt{d_{l-1}}} \mathbf{W}_l \boldsymbol{\varphi}_{l-1}(\cdot) + \mathbf{b}_l, \\ \mathbf{f}_0(\mathbf{s}; \boldsymbol{\theta}_0) &= \boldsymbol{\rho}(\mathbf{s}; \tau) \equiv (\rho_1(\mathbf{s}; \tau), \dots, \rho_K(\mathbf{s}; \tau))', \end{aligned}$$

$$\begin{aligned} \theta_{l,i}^w &\sim \text{Gau}(\mu_l^w, (\sigma_l^w)^2), \quad i = 1, \dots, n_l^w, & \theta_{l,i}^b &\sim \text{Gau}(\mu_l^b, (\sigma_l^b)^2), \quad i = 1, \dots, n_l^b, \\ \sigma_l^w &= \text{softplus}(\gamma_l^w), & \sigma_l^b &= \text{softplus}(\gamma_l^b), \end{aligned} \quad (7)$$

where $\theta_0 = \tau$. In (7), $\boldsymbol{\theta}_l^w \equiv (\theta_{l,1}^w, \dots, \theta_{l,n_l^w}^w)' = \text{vec}(\mathbf{W}_l)$ are the weight parameters at layer l ; $\mu_l^w \in \mathbb{R}$ and $\sigma_l^w \in \mathbb{R}^+$ are the prior mean and the prior standard deviation of the weights at layer l ; $\gamma_l^w \in \mathbb{R}$; and n_l^w is the number of weights in layer l . The superscript b replaces the superscript w to denote the corresponding quantities for the bias parameters $\boldsymbol{\theta}_l^b \equiv (\theta_{l,1}^b, \dots, \theta_{l,n_l^b}^b)' = \mathbf{b}_l$. The hyper-parameters that need to be estimated when calibrating the SBNN-IL are $\psi \equiv \{\{\mu_l^w, \mu_l^b, \gamma_l^w, \gamma_l^b\} : l = 1, \dots, L\}$. The spatial process $Y(\cdot)$ is the output of the final layer.

SBNN-IP: For the SBNN-I with a ‘prior-per-parameter’ scheme (SBNN-IP), the weights and biases are instead distributed as follows:

$$\begin{aligned} \text{For } l = 1, \dots, L, i = 1, \dots, n_l^w, & \quad \text{For } l = 1, \dots, L, i = 1, \dots, n_l^b, \\ \theta_{l,i}^w \sim \text{Gau}(\mu_{l,i}^w, (\sigma_{l,i}^w)^2), & \quad \theta_{l,i}^b \sim \text{Gau}(\mu_{l,i}^b, (\sigma_{l,i}^b)^2), \\ \sigma_{l,i}^w = \text{softplus}(\gamma_{l,i}^w), & \quad \sigma_{l,i}^b = \text{softplus}(\gamma_{l,i}^b). \end{aligned} \quad (8)$$

In (8), $\boldsymbol{\mu}_l^w \equiv (\mu_{l,1}^w, \dots, \mu_{l,n_l^w}^w)'$ and $\boldsymbol{\sigma}_l^w \equiv (\sigma_{l,1}^w, \dots, \sigma_{l,n_l^w}^w)'$ are the prior means and the prior standard deviations of the weights at layer l , and $\boldsymbol{\gamma}_l^w \equiv (\gamma_{l,1}^w, \dots, \gamma_{l,n_l^w}^w)'$ is a vector of real-valued elements. The superscript b replaces the superscript w to denote the corresponding quantities for the bias parameters. The hyper-parameters that need to be estimated when calibrating the SBNN-IP are $\psi \equiv \{\{\boldsymbol{\mu}_l^w, \boldsymbol{\mu}_l^b, \boldsymbol{\gamma}_l^w, \boldsymbol{\gamma}_l^b\} : l = 1, \dots, L\}$.

BNN-IL / BNN-IP: The two variants of the vanilla BNN, BNN-IL and BNN-IP, are practically identical to SBNN-IL and SBNN-IP, respectively, with the same sets of hyper-parameters ψ that need to be calibrated but with $\mathbf{f}_0(\mathbf{s}; \theta_0) = \mathbf{s}$ instead.

SBNN-VL: Denote the spatially-varying parameters in layer l of the SBNN-V at location $\mathbf{s} \in D$ as $\boldsymbol{\theta}_l(\mathbf{s}) \equiv \{\boldsymbol{\theta}_l^w(\mathbf{s}), \boldsymbol{\theta}_l^b(\mathbf{s})\}$, where $\boldsymbol{\theta}_l^w(\cdot)$ and $\boldsymbol{\theta}_l^b(\cdot)$ are the spatially-varying weight and bias parameters, respectively, and let $\boldsymbol{\theta}(\mathbf{s}) \equiv \{\boldsymbol{\theta}_l(\mathbf{s}) : l = 1, \dots, L\}$ denote all the SBNN-V parameters at location $\mathbf{s} \in D$. The SBNN-V with a ‘prior-per-layer’ scheme (SBNN-VL) is given by the following hierarchical spatial statistical model:

For $\mathbf{s} \in D$, $l = 1, \dots, L$,

$$\begin{aligned} Y(\mathbf{s}) \equiv f(\mathbf{s}; \boldsymbol{\theta}(\mathbf{s})) &= f_L(\mathbf{f}_{L-1}(\dots \mathbf{f}_1(\mathbf{f}_0(\mathbf{s}; \theta_0); \boldsymbol{\theta}_1(\mathbf{s})) \dots; \boldsymbol{\theta}_{L-1}(\mathbf{s})); \boldsymbol{\theta}_L(\mathbf{s})), \\ \mathbf{f}_l(\cdot; \boldsymbol{\theta}_l(\mathbf{s})) &= \frac{1}{\sqrt{d_{l-1}}} \mathbf{W}_l(\mathbf{s}) \boldsymbol{\varphi}_{l-1}(\cdot) + \mathbf{b}_l(\mathbf{s}), \\ \mathbf{f}_0(\mathbf{s}; \theta_0) &= \boldsymbol{\rho}(\mathbf{s}; \tau) \equiv (\rho_1(\mathbf{s}; \tau), \dots, \rho_K(\mathbf{s}; \tau))', \end{aligned}$$

$$\begin{aligned} \theta_{l,i}^w(\mathbf{s}) &= \mu_l^w(\mathbf{s}) + \sigma_l^w(\mathbf{s}) \eta_{l,i}^w, & \theta_{l,i}^b(\mathbf{s}) &= \mu_l^b(\mathbf{s}) + \sigma_l^b(\mathbf{s}) \eta_{l,i}^b, & i &= 1, \dots, n_l^w, \\ \mu_l^w(\mathbf{s}) &= (\boldsymbol{\alpha}_l^w)' \boldsymbol{\rho}(\mathbf{s}), & \mu_l^b(\mathbf{s}) &= (\boldsymbol{\alpha}_l^b)' \boldsymbol{\rho}(\mathbf{s}), & & \\ \sigma_l^w(\mathbf{s}) &= \text{softplus}((\boldsymbol{\beta}_l^w)' \boldsymbol{\rho}(\mathbf{s})), & \sigma_l^b(\mathbf{s}) &= \text{softplus}((\boldsymbol{\beta}_l^b)' \boldsymbol{\rho}(\mathbf{s})), & & \\ \eta_{l,i}^w &\stackrel{\text{iid}}{\sim} \text{Gau}(0, 1), & \eta_{l,i}^b &\stackrel{\text{iid}}{\sim} \text{Gau}(0, 1), & i &= 1, \dots, n_l^b, \end{aligned} \quad (9)$$

where $\theta_0 = \tau$. In (9), $\boldsymbol{\theta}_l^w(\cdot) \equiv (\theta_{l,1}^w(\cdot), \dots, \theta_{l,n_l^w}^w(\cdot))' = \text{vec}(\mathbf{W}_l(\cdot))$ are the weight parameters at layer l ; $\boldsymbol{\alpha}_l^w \equiv (\alpha_{l,1}^w, \dots, \alpha_{l,K}^w)'$ and $\boldsymbol{\beta}_l^w \equiv (\beta_{l,1}^w, \dots, \beta_{l,K}^w)'$ are vectors of real-valued basis-function coefficients for the prior mean and the prior standard deviation of the weights at layer l ; and n_l^w is the number of weights in layer l . The superscript b replaces the superscript w to denote the corresponding quantities for the spatially-varying bias parameters

$\boldsymbol{\theta}_l^b(\cdot) \equiv (\theta_{l,1}^b(\cdot), \dots, \theta_{l,n_l^b}^b(\cdot))' = \mathbf{b}_l(\cdot)$. The hyper-parameters that need to be estimated when calibrating the SBNN-VL are $\psi \equiv \{\{\boldsymbol{\alpha}_l^w, \boldsymbol{\alpha}_l^b, \boldsymbol{\beta}_l^w, \boldsymbol{\beta}_l^b\} : l = 1, \dots, L\}$. As with the SBNN-IL, the spatial process $Y(\cdot)$ is the output of the final layer.

SBNN-VP: For the SBNN-V with a ‘prior-per-parameter’ scheme (SBNN-VP), the weights and biases are instead given by:

$$\begin{aligned} \text{For } \mathbf{s} \in D, l = 1, \dots, L, i = 1, \dots, n_l^w, & \quad \text{For } \mathbf{s} \in D, l = 1, \dots, L, i = 1, \dots, n_l^b, \\ \theta_{l,i}^w(\mathbf{s}) = \mu_{l,i}^w(\mathbf{s}) + \sigma_{l,i}^w(\mathbf{s})\eta_{l,i}^w, & \quad \theta_{l,i}^b(\mathbf{s}) = \mu_{l,i}^b(\mathbf{s}) + \sigma_{l,i}^b(\mathbf{s})\eta_{l,i}^b, \\ \mu_{l,i}^w(\mathbf{s}) = (\boldsymbol{\alpha}_{l,i}^w)' \boldsymbol{\rho}(\mathbf{s}), & \quad \mu_{l,i}^b(\mathbf{s}) = (\boldsymbol{\alpha}_{l,i}^b)' \boldsymbol{\rho}(\mathbf{s}), \\ \sigma_{l,i}^w(\mathbf{s}) = \text{softplus}((\boldsymbol{\beta}_{l,i}^w)' \boldsymbol{\rho}(\mathbf{s})), & \quad \sigma_{l,i}^b(\mathbf{s}) = \text{softplus}((\boldsymbol{\beta}_{l,i}^b)' \boldsymbol{\rho}(\mathbf{s})), \\ \eta_{l,i}^w \stackrel{\text{iid}}{\sim} \text{Gau}(0, 1), & \quad \eta_{l,i}^b \stackrel{\text{iid}}{\sim} \text{Gau}(0, 1). \end{aligned} \quad (10)$$

In (10), $\boldsymbol{\alpha}_{l,i}^w \equiv (\alpha_{l,i,1}^w, \dots, \alpha_{l,i,K}^w)'$ and $\boldsymbol{\beta}_{l,i}^w \equiv (\beta_{l,i,1}^w, \dots, \beta_{l,i,K}^w)'$ are real-valued basis-function coefficients for the prior mean and the prior standard deviation of the i th weight at layer l . The superscript b replaces the superscript w to denote the corresponding quantities for the spatially-varying bias parameters. The hyper-parameters that need to be estimated when calibrating the SBNN-VP are $\psi \equiv \{\{\mathbf{A}_l^w, \mathbf{A}_l^b, \mathbf{B}_l^w, \mathbf{B}_l^b\} : l = 1, \dots, L\}$, where matrix $\mathbf{A}_l^w \equiv (\boldsymbol{\alpha}_{l,1}^w, \dots, \boldsymbol{\alpha}_{l,n_l^w}^w)$, with $\mathbf{A}_l^b, \mathbf{B}_l^w$ and \mathbf{B}_l^b similarly defined.

3. SBNN calibration using the Wasserstein distance

Assume now that we have access to realisations from a second stochastic process over D , $\tilde{Y}(\cdot)$, which we refer to as the target process. These realisations from $\tilde{Y}(\cdot)$ could be simulations from a stochastic simulator or data from a re-analysis product, say. Assume further that we want an SBNN to have a selected finite-dimensional distribution that ‘matches’ that of the target’s, in a sense detailed below. In this section we outline an approach to adjusting the SBNN hyper-parameters ψ in order to achieve this objective; we refer to the procedure of choosing ψ such that a given finite-dimensional distribution of the two processes are matched as ‘calibration.’ Calibration is a difficult task as it involves exploring the space of high-dimensional distribution functions, and was not considered computationally feasible until very recently. The method we adopt for calibration, detailed in [Tran et al. \(2022\)](#), makes the problem computationally tractable. Their method is based on minimising a Wasserstein distance (see, e.g., [Panaretos and Zemel, 2019](#)) between the given finite-dimensional distributions of the two processes via Monte Carlo approximations.

Consider a matrix of locations $\mathbf{S} \equiv (\mathbf{s}_1, \dots, \mathbf{s}_n)$ where $\mathbf{s}_1, \dots, \mathbf{s}_n \in D$ with $n \geq 1$ (typically \mathbf{S} is a fine gridding of D). For the SBNN-V, $\Theta \equiv \{\boldsymbol{\theta}(\mathbf{s}_1), \dots, \boldsymbol{\theta}(\mathbf{s}_n)\}$ are the parameters at the set of locations, while $\mathbf{Y} \equiv (Y(\mathbf{s}_1), \dots, Y(\mathbf{s}_n))' = (f(\mathbf{s}_1; \boldsymbol{\theta}(\mathbf{s}_1)), \dots, f(\mathbf{s}_n; \boldsymbol{\theta}(\mathbf{s}_n)))' \equiv \mathbf{f}_\Theta$ denotes a vector whose elements are the process evaluated at these locations. The same definitions hold for the SBNN-I variants, but with $\boldsymbol{\theta}(\mathbf{s}_i)$ replaced with $\boldsymbol{\theta}$, for $i = 1, \dots, n$. The distribution of \mathbf{Y} is analytically intractable, but it is straightforward to simulate from, since $p(\boldsymbol{\Theta}; \psi)$ (the prior distribution of the weights and biases) is straightforward to simulate from (see, e.g., (6)) and $\mathbf{Y} = \mathbf{f}_\Theta$ is a deterministic function of $\boldsymbol{\Theta}$ (see (1)). We collect the corresponding evaluations of the second, or target, process $\tilde{Y}(\cdot)$ in $\tilde{\mathbf{Y}} \equiv (\tilde{Y}(\mathbf{s}_1), \dots, \tilde{Y}(\mathbf{s}_n))'$. We then match the distribution of \mathbf{Y} to the empirical distribution of $\tilde{\mathbf{Y}}$ by minimising the dissimilarity between these two distributions. The natural choice of dissimilarity measure to minimise is the Kullback-Leibler divergence; however, this divergence term contains an entropy term that is analytically intractable and computationally difficult to approximate ([Flam-Shepherd et al., 2017](#); [Delattre and Fournier, 2017](#)). On the other hand, the Wasserstein distance leads to no such difficulty.

The Wasserstein distance is a measure of dissimilarity between two probability distributions. As in [Tran et al. \(2022\)](#), we consider a special case of the metric, the *Wasserstein-1 distance* that is given by

$$W_1(\psi) \equiv \sup_{\phi: \|\nabla \phi\| \leq 1} \mathbb{E}[\phi(\mathbf{Y}) \mid \psi] - \mathbb{E}[\phi(\tilde{\mathbf{Y}})], \quad (11)$$

where the first expectation is taken with respect to the distribution derived from the SBNN; the second expectation is taken with respect to the distribution derived from the target process; $\phi(\cdot)$ is differentiable; and where the constraint on the gradient norm $\|\nabla \phi\| \leq 1$ ensures that $\phi(\cdot)$ is also 1-Lipschitz. A Monte Carlo approximation to (11) yields the empirical expectations,

$$W_1(\psi) \approx \sup_{\phi: \|\nabla \phi\| \leq 1} \frac{1}{N} \sum_{i=1}^N \left(\phi(\mathbf{Y}_\psi^{(i)}) - \phi(\tilde{\mathbf{Y}}^{(i)}) \right), \quad (12)$$

where $\mathbf{Y}_\psi^{(i)} \equiv \mathbf{f}_{\Theta^{(i)}} \text{ for } \Theta^{(i)} \sim p(\Theta; \psi)$, $\tilde{\mathbf{Y}}^{(i)}$ is a sample from the target process, and N is the number of samples from each process. Note that one could choose different sample sizes for the Monte Carlo approximation of the two expectations in (11) if needed; for ease of exposition we assume that both sample sizes are equal to N . Calibration proceeds by minimising this approximate Wasserstein distance. Specifically, we seek ψ^* , where

$$\psi^* = \arg \min_{\psi} W_1(\psi). \quad (13)$$

Since all the functions in the BNN are differentiable with respect to ψ , we use gradient descent to carry out the optimisation in a computationally efficient way.

The Wasserstein distance in (11) is a supremum over the space of 1-Lipschitz functions. There is no computationally feasible way to explore this function space and, therefore, we adopt the same approach as in [Tran et al. \(2022\)](#) and model $\phi(\cdot)$ using a neural network parameterised by some parameters λ . That is, we replace $\phi(\cdot)$ with a neural network $\phi_{NN}(\cdot; \lambda)$ in (11), which is differentiable by construction. This leads to the following approximation of the Wasserstein-1 distance (11),

$$W_1(\psi) \approx \mathbb{E}[\phi_{NN}(\mathbf{Y}; \lambda^*) \mid \psi] - \mathbb{E}[\phi_{NN}(\tilde{\mathbf{Y}}; \lambda^*)],$$

where the optimised neural network parameters are given by

$$\lambda^* = \arg \max_{\lambda} \mathbb{E}[\phi_{NN}(\mathbf{Y}; \lambda) \mid \psi] - \mathbb{E}[\phi_{NN}(\tilde{\mathbf{Y}}; \lambda)] \quad (14)$$

$$\approx \arg \max_{\lambda} \frac{1}{N} \sum_{i=1}^N \left(\phi_{NN}(\mathbf{Y}_\psi^{(i)}; \lambda) - \phi_{NN}(\tilde{\mathbf{Y}}^{(i)}; \lambda) \right), \quad (15)$$

where we have again chosen to use the sample size N for the approximation of both expectations for notational convenience.

So far we have not ensured that $\|\nabla \phi_{NN}(\cdot; \lambda^*)\| \leq 1$ (i.e., 1-Lipschitz continuity), just that $\phi_{NN}(\cdot; \lambda^*)$ is differentiable (by construction). To ensure that $\phi_{NN}(\cdot; \lambda^*)$ is at least approximately 1-Lipschitz, we make use of a remarkable result by [Gulrajani et al. \(2017\)](#), which we summarise as follows: For the weighted average $\bar{\mathbf{Y}} \equiv \delta \mathbf{Y} + (1 - \delta) \tilde{\mathbf{Y}}$, $0 \leq \delta \leq 1$, the 1-Lipschitz function that maximises (11), $\phi^*(\cdot)$ say, has gradient-norm $\|\nabla \phi^*(\bar{\mathbf{Y}})\| = 1$ with probability one. Therefore, when exploring the space of differentiable functions, one need only consider those functions that have a gradient-norm at $\bar{\mathbf{Y}}$ equal to one. In practice, we can use this result to

suggest a penalisation term to add to (15); specifically, we alter (14) and (15) as follows:

$$\boldsymbol{\lambda}^* = \arg \max_{\boldsymbol{\lambda}} \mathbb{E}[\phi_{NN}(\mathbf{Y}; \boldsymbol{\lambda}) | \boldsymbol{\psi}] - \mathbb{E}[\phi_{NN}(\tilde{\mathbf{Y}}; \boldsymbol{\lambda})] - \zeta \mathbb{E}\left[\left(\|\nabla \phi_{NN}(\bar{\mathbf{Y}}; \boldsymbol{\lambda})\| - 1\right)^2\right] \quad (16)$$

$$\approx \arg \max_{\boldsymbol{\lambda}} \frac{1}{N} \sum_{i=1}^N \left(\phi_{NN}(\mathbf{Y}_{\boldsymbol{\psi}}^{(i)}; \boldsymbol{\lambda}) - \phi_{NN}(\tilde{\mathbf{Y}}^{(i)}; \boldsymbol{\lambda}) - \zeta \left(\|\nabla \phi_{NN}(\bar{\mathbf{Y}}^{(i)}; \boldsymbol{\lambda})\| - 1\right)^2 \right), \quad (17)$$

where, for $i = 1, \dots, N$, $\bar{\mathbf{Y}}^{(i)} = \delta^{(i)} \mathbf{Y}^{(i)} + (1 - \delta^{(i)}) \tilde{\mathbf{Y}}^{(i)}$, $\delta^{(i)}$ is independently drawn from the bounded uniform distribution on $[0, 1]$, and ζ is a tuning parameter. Following the recommendation of Gulrajani et al. (2017) we set $\zeta = 10$. Note that this “soft-constraint” approach does not guarantee that $\phi_{NN}(\cdot; \boldsymbol{\lambda}^*)$ will be 1-Lipschitz. However, our results in Section 4 suggest that this approach is effective. Neural networks that enforce 1-Lipschitz continuity by design are available, and we discuss them in Section 6.

Note that (13) and (16) together form a two-stage optimisation problem, which we solve using gradient methods. In the first stage, which we refer to as the inner-loop optimisation, we optimise $\boldsymbol{\lambda}$ using gradient ascent while keeping $\boldsymbol{\psi}$ fixed, in order to establish the Wasserstein metric for the fixed value of $\boldsymbol{\psi}$. In the second stage, which we refer to as the outer-loop optimisation, we do a single gradient descent step to find a new $\boldsymbol{\psi}$ (conditional on $\boldsymbol{\lambda}$) that reduces the Wasserstein distance. We only do one step at a time in the outer-loop optimisation stage since the Wasserstein metric needs to be re-established for every new value of $\boldsymbol{\psi}$. We generate N samples from \mathbf{Y} , $\tilde{\mathbf{Y}}$, and $\bar{\mathbf{Y}}$, once for each inner-loop optimisation and once for each outer-loop step. We iterate both stages until the Wasserstein distance ceases to notably decrease after several outer-loop optimisation steps.

The calibration procedure outlined above optimises $\boldsymbol{\psi}$ such that the distribution of \mathbf{Y} (i.e., $Y(\cdot)$ evaluated over \mathbf{S}) is close, in a Wasserstein-1 sense, to that of $\tilde{\mathbf{Y}}$ (i.e., $\tilde{Y}(\cdot)$ evaluated over the matrix of locations \mathbf{S}). The choice of \mathbf{S} determines the finite-dimensional distribution being compared. In applications where we have access to the realisations $\tilde{\mathbf{Y}}$ but where we cannot simulate from $\tilde{Y}(\cdot)$ at arbitrary locations, \mathbf{S} is fixed by the application. When we can freely simulate from $\tilde{Y}(\cdot)$, then, since d is typically small in spatial applications, one may define \mathbf{S} as a fine gridding over D , and this is the approach we present throughout this paper. The optimised hyper-parameters $\boldsymbol{\psi}^*$ lead to an SBNN, $Y(\cdot)$, that approximates the target process $\tilde{Y}(\cdot)$ well over the locations in \mathbf{S} .

4. Simulation studies

In all the simulation studies given in this section, we consider the special case that the target process $\tilde{Y}(\cdot)$ is known and that it can be easily simulated from, so that we can compare the (S)BNNs to the process to which they are calibrated. In practice, the target process may be unknown, but to use our approach we would need realisations from it that are used to calibrate the (S)BNNs. We consider three processes for $\tilde{Y}(\cdot)$: a stationary Gaussian spatial process (Section 4.1); a non-stationary Gaussian spatial process (Section 4.2); and a stationary lognormal spatial process (Section 4.3). We define all these processes on a two-dimensional spatial domain $D = [-4, 4] \times [-4, 4]$, and we set $\mathbf{s}_1, \dots, \mathbf{s}_n$ as the centroids of a 64×64 gridding of D (so that \mathbf{S} is a matrix of size 2×4096). For the activation functions in $\varphi_l(\cdot)$, $l = 0, \dots, L-1$, we use the $\tanh(\cdot)$ function, where $\tanh : t \mapsto (e^{2t} - 1)/(e^{2t} + 1)$. We use $L = 4$ layers and set each hidden layer to have dimension 40; that is, we set $d_1 = d_2 = d_3 = 40$. The input and output dimensions are $d = 2$ and $d_4 = 1$, while $d_0 = 2$ for vanilla BNNs, and $d_0 = K$ for SBNNs, where K is the number of basis functions in the embedding layer. We let $\phi_{NN}(\cdot; \boldsymbol{\lambda})$ be a neural network with two hidden layers each of the form (2) with dimension 200 and with softplus activation functions. The weights and biases of the two layers, which we collect in $\boldsymbol{\lambda}$,

are initialised by simulating from the bounded uniform distribution on $[-\sqrt{z}, \sqrt{z}]$, where z is given by the reciprocal of the input dimension of the respective layer (He et al., 2015).

Throughout the simulation studies we consider the six (S)BNNs discussed in Section 2.2.3. For the SBNNs we set $\mathbf{f}_0(\cdot; \theta_0) = \boldsymbol{\rho}(\cdot; \tau)$ as in (3) with length scale $\theta_0 = \tau = 1$, and we use $K = 15^2$ radial basis functions arranged on a 15×15 grid in D . In Figure S1 in the Supplementary Material we show evaluations of the basis functions, which have their centroid at $s_1 = 0$, over $\mathbf{s} = (0, s_2)'$, where $s_2 \in [-4, 4]$. We did not conduct a detailed experiment to analyse how the results change with τ ; provided there is reasonable overlap between the basis functions, as in our case, we do not expect the results to substantially change with τ .

As initialisation for our six models, we set all the μ 's equal to zero, and all the γ 's equal to one for the BNN-I and SBNN-I variants. For the SBNN-V variants, we set all the α 's equal to zero, while we simulate all the β 's independently from a normal distribution with zero mean and unit variance. When calibrating, we sample N parameter vectors, $\{\boldsymbol{\Theta}^{(i)} : i = 1, \dots, N\}$, from the prior on the weights and biases, N corresponding realisations from the (S)BNN $\{\mathbf{Y}^{(i)} : i = 1, \dots, N\}$, and N realisations from the target process $\{\tilde{\mathbf{Y}}^{(i)} : i = 1, \dots, N\}$, where $N = 1024$ for the BNN-I and SBNN-I variants, and where $N = 512$ for the SBNN-V variants, in order to reduce memory requirements. Using these simulations, we then carry out 50 gradient steps when optimising $\boldsymbol{\lambda}$ and keeping $\boldsymbol{\psi}$ fixed in (15) (inner-loop optimisation). We then re-simulate to obtain N realisations of $Y(\cdot)$ and $\tilde{Y}(\cdot)$ before carrying out a single gradient step when optimising $\boldsymbol{\psi}$ and keeping $\boldsymbol{\lambda}$ fixed in (12) (outer-loop optimisation). Recall that we only do a single gradient step when optimising $\boldsymbol{\psi}$ as the Wasserstein metric in (11) depends on $\boldsymbol{\psi}$, and thus needs to be re-established (i.e., $\boldsymbol{\lambda}$ needs to be re-estimated) for every update of $\boldsymbol{\psi}$. We repeat this two-stage procedure of iteratively optimising $\boldsymbol{\lambda}$ and $\boldsymbol{\psi}$ (always using the most recently updated values of $\boldsymbol{\lambda}$ and $\boldsymbol{\psi}$ as initial conditions) until the Wasserstein distance stabilises. Since we are generating data ‘on-the-fly’ when calibrating, there is little risk of over-fitting (Chan et al., 2018); see Figure S2 in the Supplementary Material for plots of the Wasserstein distances for all models and simulation experiments as a function of the outer-loop optimisation step. Note that these Wasserstein distances are necessarily approximate since the 1-Lipschitz functions are approximated using a neural network, as outlined in Section 3.

As in Tran et al. (2022), we use Adagrad and RMSprop (Kochenderfer and Wheeler, 2019) strategies for adjusting the gradient step sizes in the inner- and outer-loop optimisations, respectively.

In Table 1 we show the Wasserstein-1 distance averaged over the final 100 outer-loop iterations for the six (S)BNNs we consider. In this table we also list the number of hyper-parameters associated with each model. In the case of the ‘prior-per-layer’ BNN-IL and SBNN-IL, the number of hyper-parameters is 16 (two mean and two variance hyper-parameters for each layer). In the case of the ‘prior-per-layer’ SBNN-VL, the number of hyper-parameters in each of the four layers is $4K = 900$, for a total of 3600 hyper-parameters. The ‘prior-per-parameter’ models are considerably more parameterised since, for these models, the number of hyper-parameters grows linearly with the number of weights and biases in the model. The number of weights and biases is 3441 for BNN-IL and BNN-IP, and it is 12361 for all the SBNNs with an embedding layer. Multiplying the number of parameters by two for the BNN-I and the SBNN-I variants, and by $2K$ for the SBNN-V variant, gives the total number of hyper-parameters associated with each of these ‘prior-per-parameter’ models. A cursory glance at this table reveals that a larger number of hyper-parameters generally, but not always, leads to a lower Wasserstein distance, and that the SBNN variants generally outperform the BNN-I variants, sometimes by a substantial margin. We explore the nuances in more detail in the following sections.

Reproducible code, which contains additional details on the simulation studies in this section, is available from <https://github.com/andrewzm/SBNN>.

Table 1: Number of parameters, number of hyper-parameters, and Wasserstein distance on convergence for each simulation experiment and (S)BNN combination. The reported Wasserstein distance is computed as the average over the distances in the final 100 outer-loop optimisation steps.

Model	Num. par.	Num. hyper-par.	$W_1(\psi)$ (Stat. GP)	$W_1(\psi)$ (Non-stat. GP)	$W_1(\psi)$ (Log-GP)
BNN-IL	3441	16	107.14	40.37	28.18
BNN-IP	3441	6882	122.33	60.35	9.09
SBNN-IL	12361	16	4.22	95.64	10.67
SBNN-IP	12361	24722	6.72	2.65	0.69
SBNN-VL	12361	3600	7.86	6.70	8.37
SBNN-VP	12361	5562450	2.00	0.95	1.03

4.1. Calibration to stationary Gaussian spatial process

In this simulation study we consider a mean zero stationary and isotropic Gaussian spatial process with unit variance and squared-exponential covariance function as our target process $\tilde{Y}(\cdot)$. Hence, $\tilde{\mathbf{Y}} \sim \text{Gau}(\mathbf{0}, \Sigma)$, where $\Sigma \equiv (\text{cov}(\tilde{Y}(\mathbf{s}_k), \tilde{Y}(\mathbf{s}_l)) : k, l = 1, \dots, n)$. We model the covariances through a squared-exponential covariogram $C^o(\cdot)$,

$$\text{cov}(\tilde{Y}(\mathbf{s}), \tilde{Y}(\mathbf{r})) \equiv C^o(\|\mathbf{s} - \mathbf{r}\|) = \exp\left(-\frac{\|\mathbf{s} - \mathbf{r}\|^2}{2\ell^2}\right), \quad \mathbf{s}, \mathbf{r} \in D, \quad (18)$$

where we set the length scale $\ell = 1$. From Table 1 we see that all the SBNN variants perform similarly in this case, and considerably outperform the BNN-I variants. For this reason, in the following discussion we focus on comparing the calibrated BNN-IL and the calibrated SBNN-IL to the target Gaussian process; the shown results are representative of those from their respective variants.

We first compare the two models through the empirical covariogram calculated using samples from the networks. Rather than only computing the empirical covariogram at the final (in this case, the 4000th) optimisation step, we compute empirical estimates at a number of intermediate steps in order to monitor the adaptation of the (S)BNN to the target process during optimisation. Specifically, we compute empirical covariograms after 100, 200, 400, 2000, and 4000 outer-loop gradient steps, respectively, and compare these estimates to the true covariogram of the target Gaussian process. Figure 3, left panel, shows that the calibrated BNN-IL fails to recover the true covariogram; on convergence, BNN-IL’s covariogram has smaller intercept at the origin and slower decrease as the spatial lag increases. On the other hand, Figure 3, right panel, shows that the covariogram of the SBNN-IL with an embedding layer converges (after approximately 2000 outer-loop gradient steps) to a covariogram that is very similar to that of the target process.

In Figure 4, top-left panel, we plot heat maps of the target process’ covariances, $\text{cov}(\tilde{Y}(\mathbf{s}_0), \tilde{Y}(\mathbf{S}))$, for 16 values of \mathbf{s}_0 arranged on a regular 4×4 grid in D , where recall that \mathbf{S} is made up of the grid-cell centroids of a 64×64 gridding of D . In the bottom-left panel we plot realisations of $\tilde{\mathbf{Y}}$. In the top-right panel we show the corresponding empirical estimates of the covariances $\text{cov}(Y(\mathbf{s}_0), Y(\mathbf{S}))$ from the calibrated SBNN-IL. These covariances are indicative of stationarity and isotropy, and they are very similar to those of the target process. This is reassuring as there is nothing in the construction or training procedure of the SBNN-IL that constrains the process to be stationary or isotropic; the similarity between the covariances is another indication that the SBNN-IL is targeting the correct process. In the bottom-right panel of Figure 4 we plot sample realisations of \mathbf{Y} . These realisations have very similar properties to those of $\tilde{\mathbf{Y}}$ (similar length scale, smoothness, and variance).

Although the SBNN-IL is a highly non-Gaussian process, it has been calibrated against a Gaussian process, and thus all its finite-dimensional distributions derived from the high-

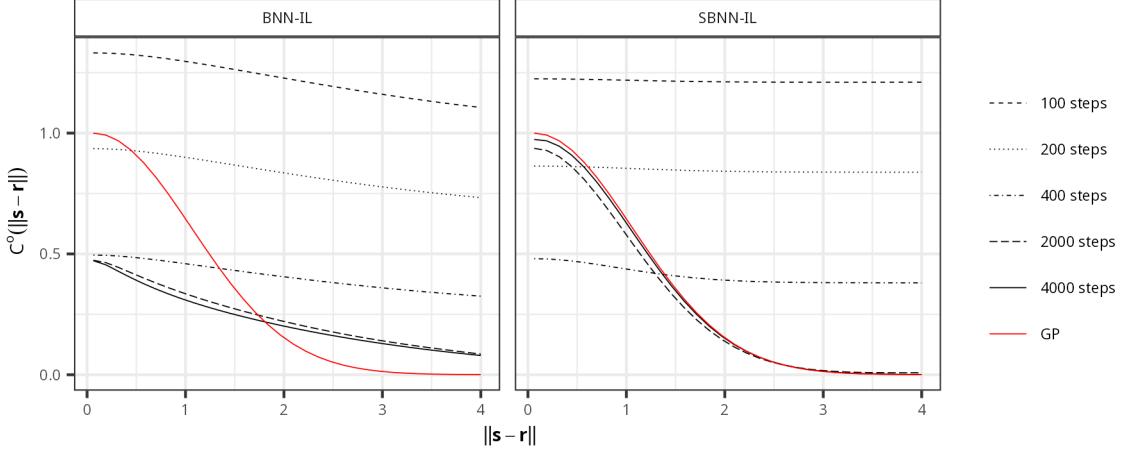


Fig. 3: Empirical covariogram of the (S)BNN at different stages during the optimisation (different line styles denote the empirical covariogram after 100, 200, 400, 2000, and 4000 gradient steps, respectively), and the true covariogram of the target Gaussian process (red). Left: BNN-IL. Right: SBNN-IL.

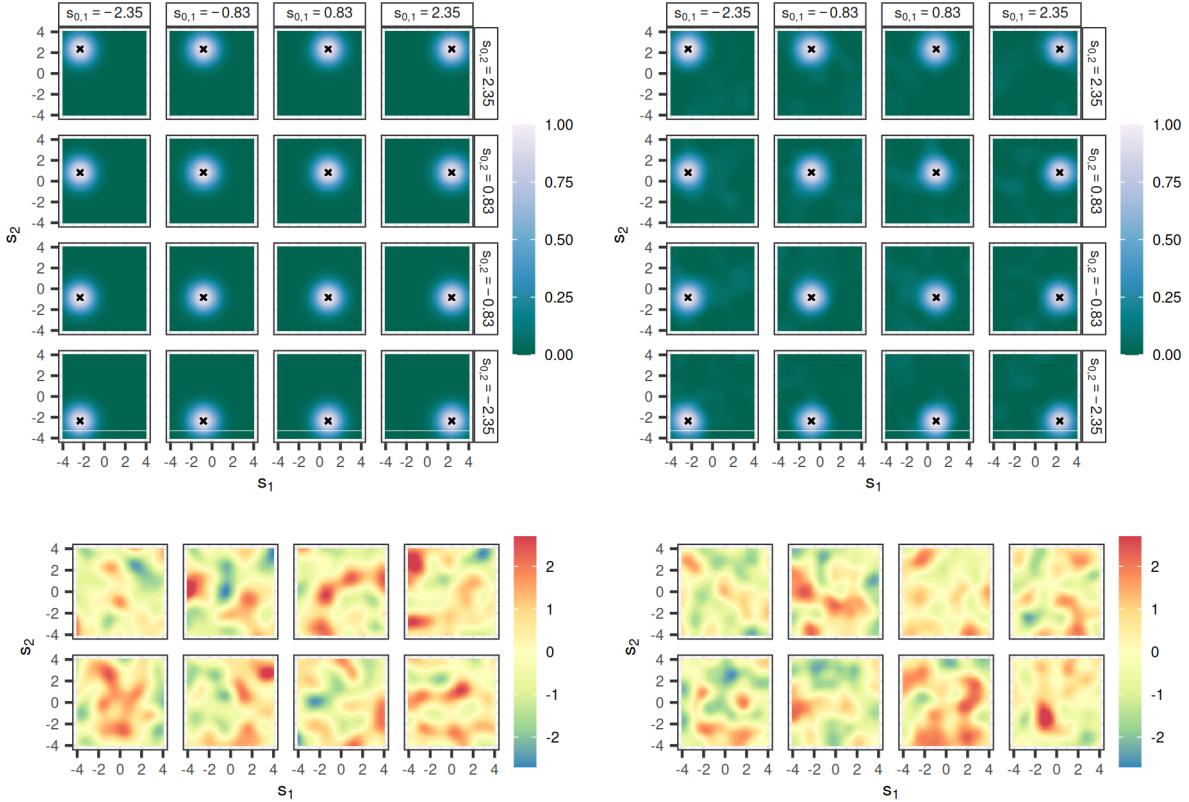


Fig. 4: (Top-left panel) Covariance between the target process at 16 grid points (crosses) with coordinates as indicated by the labels at the top and the right of the sub-panels, and the target process on a fine gridding of D . (Bottom-left panel) Eight realisations of the underlying target process on a fine gridding of D . (All right panels) Same as left panels but for the calibrated SBNN-IL.

dimensional one that was used for calibration should also be Gaussian. To illustrate that Gaussianity is well approximated, in Figure 5 we plot kernel density estimates from 1000 samples taken from the calibrated SBNN-IL and the true Gaussian process. The top panel shows empirical marginal densities of $Y(\mathbf{s}_0)$ and $\tilde{Y}(\mathbf{s}_0)$ for eight values of \mathbf{s}_0 arranged on a 2×4 grid

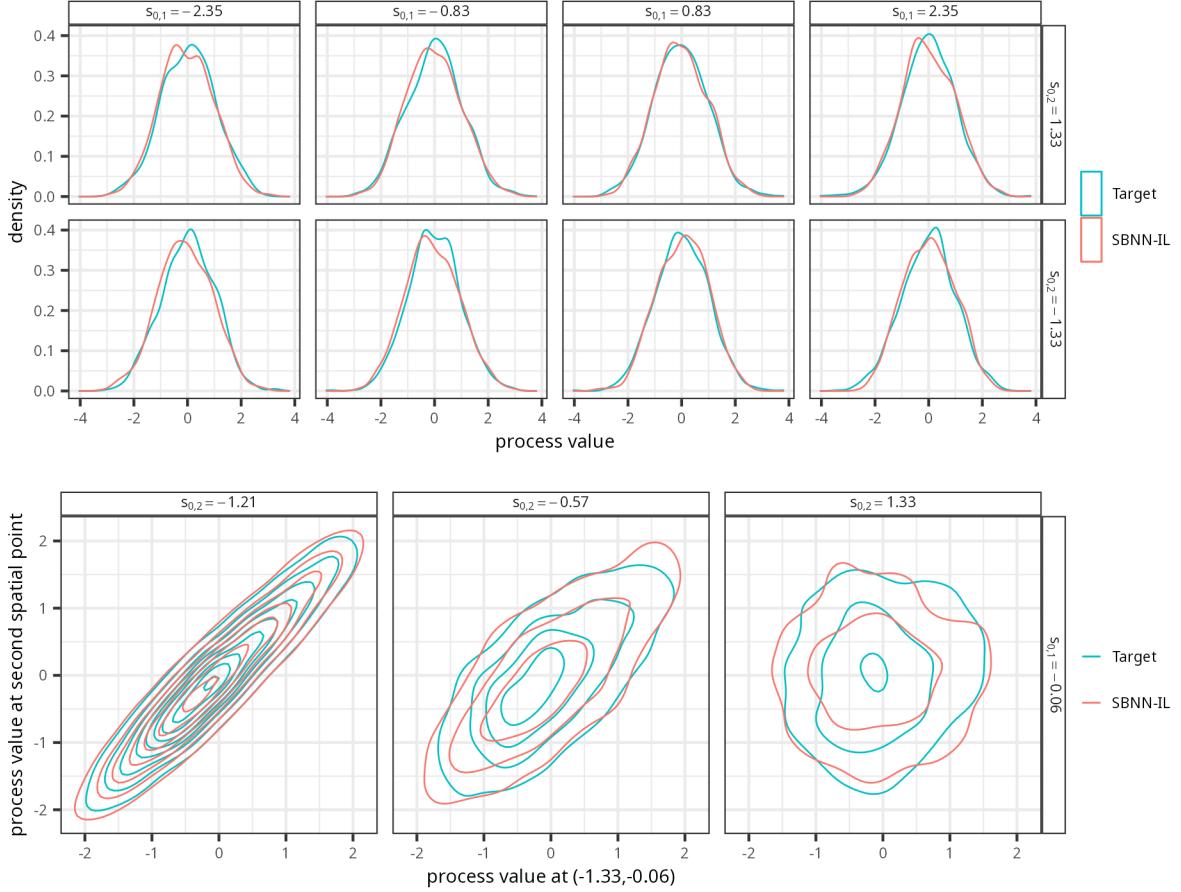


Fig. 5: Kernel density plots from 1000 samples of the SBNN-IL and the target stationary Gaussian spatial process. (Top panel) Univariate densities of the two processes at eight spatial points arranged on a 2×4 grid in D (with coordinates as indicated by the labels of the sub-panels). (Bottom panel) Bivariate densities of the two processes at $\tilde{s}_0 = (-1.33, -0.06)'$ and three other points in D (with coordinates as indicated by the labels of the sub-panels). Contour levels (from the outer to the inner contours) correspond to 0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40 and 0.45.

in D , while the bottom panel shows bivariate densities corresponding to $(Y(\tilde{s}_0), Y(s_0))'$ and $(\tilde{Y}(\tilde{s}_0), \tilde{Y}(s_0))'$, for $\tilde{s}_0 = (-1.33, -0.06)'$ and three choices for the location s_0 : One pair of coordinates is close to \tilde{s}_0 (left sub-panel); one pair is far from \tilde{s}_0 (right sub-panel); and the last pair is in between these first two (middle sub-panel). Both the marginal and the joint densities are very similar and suggest that Gaussianity of the finite-dimensional distribution has been well-approximated during calibration. Overall, the evidence points to the calibrated SBNN-IL being a very good approximation to the underlying Gaussian process.

In Figures S3, S4, and S5 in the Supplementary Material, we show the corresponding figures for the calibrated SBNN-VL, which clearly also gives a very good approximation to the underlying process. These results are reassuring as they suggest that the SBNN-VL can model stationary processes well, despite its extra complexity that was largely introduced to model non-stationary processes.

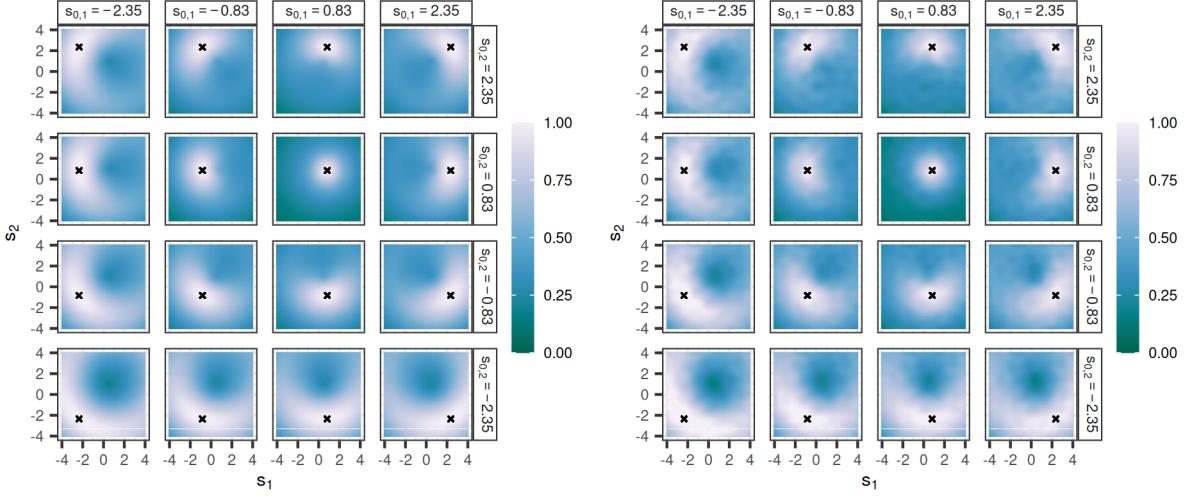


Fig. 6: (Left panel) Covariance between the target non-stationary Gaussian spatial process at 16 grid points (crosses) arranged on a 4×4 grid (corresponding to the 16 sub-panels) and the process on a fine gridding of D . (Right panel) Same as left panel, but for the calibrated SBNN-VL.

4.2. Calibration to non-stationary Gaussian spatial process

In this simulation study we consider a non-stationary Gaussian process with mean zero, unit variance, and covariance function

$$C(\mathbf{s}, \mathbf{r}) = |\Sigma(\mathbf{s})|^{1/4} |\Sigma(\mathbf{r})|^{1/4} \left| \frac{\Sigma(\mathbf{s}) + \Sigma(\mathbf{r})}{2} \right|^{-1/2} C^o\left(\sqrt{Q(\mathbf{s}, \mathbf{r})}\right), \quad \mathbf{s}, \mathbf{r} \in D,$$

where $C^o(\cdot)$ is the squared-exponential covariogram defined in (18). The matrix $\Sigma(\cdot) \in \mathbb{R}^{d \times d}$ is the so-called kernel matrix (Paciorek and Schervish, 2006), and $\sqrt{Q(\mathbf{s}, \mathbf{r})}$ is the inter-point Mahalanobis distance, where

$$Q(\mathbf{s}, \mathbf{r}) = (\mathbf{s} - \mathbf{r})^\top \left(\frac{\Sigma(\mathbf{s}) + \Sigma(\mathbf{r})}{2} \right)^{-1} (\mathbf{s} - \mathbf{r}).$$

We let the kernel matrix $\Sigma(\mathbf{s}) = \exp(\kappa \|\mathbf{s} - \boldsymbol{\xi}\|) \mathbf{I}$, $\mathbf{s} \in D$, where $\boldsymbol{\xi} = (0.5, 1)'$ is a point in D , and here we set the scaling parameter $\kappa = 1$. From Table 1 we see that the first three models, BNN-IL, BNN-IP, and SBNN-IL, when calibrated do not match the target process as well as the other SBNN variants.

In Figure 6 we plot the true covariance function of the target Gaussian process (left panel), and the empirical covariance function of the calibrated SBNN-VL (right panel), which is very similar to that of the calibrated SBNN-IP (not shown) and that of the calibrated SBNN-VP (not shown). Each heatmap in these panels depicts covariances of the process with respect to the process at a specific spatial location (denoted by the cross). The covariance structure “revolves” around the point $\boldsymbol{\xi} = (0.5, 1)'$, so that it is approximately isotropic near this point, and anisotropic away from the centre. The SBNN-VL clearly captures this covariance structure. The BNN-IL and SBNN-IL, whose analogous plots are shown in Figure S6 in the Supplementary Material, clearly do not. The SBNN-IL provides a particularly poor fit to the covariance, suggesting that it should be only used to model stationary processes. In Figure 7 we plot multiple sample paths from the target process and from all the calibrated models. The sample paths from the calibrated SBNN-VP are clearly very similar to those of the target process, while those from the calibrated BNN-IL and BNN-IP and the calibrated SBNN-IL are noticeably different. Conclusions from the sample paths match those given above for the covariance functions.

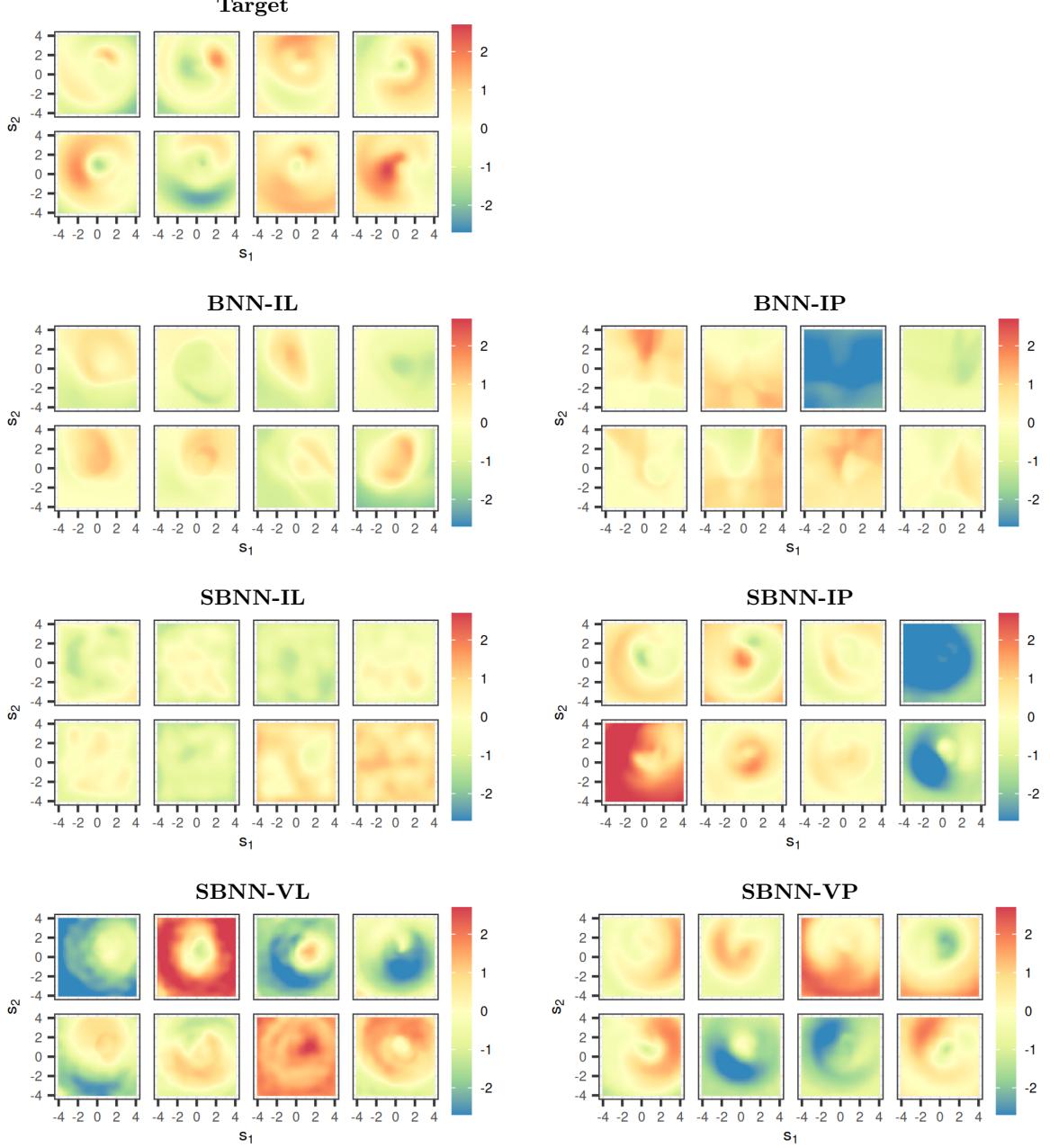


Fig. 7: Samples from the target, non-stationary, Gaussian process and all six (S)BNN variants calibrated to the target process.

4.3. Calibration to lognormal spatial process

We now consider a target stationary lognormal spatial process, specifically a process $\tilde{Y}(\cdot)$ where $\log(\tilde{Y}(\cdot))$ is a stationary Gaussian spatial process. Hence, $\tilde{\mathbf{Y}} \sim \text{LN}(\mathbf{0}, \Sigma)$, where $\mathbf{X} \sim \text{LN}(\boldsymbol{\mu}, \mathbf{C})$ denotes a multivariate lognormal distribution with $\mathbb{E}[\log(\mathbf{X})] = \boldsymbol{\mu}$ and $\text{Var}[\log(\mathbf{X})] = \mathbf{C}$. In order to demonstrate that our SBNNs can also effectively model processes that are not smooth, we specify the target process covariances using a Matérn 3/2 covariogram. That is,

$$\text{cov}(\tilde{Y}(\mathbf{s}), \tilde{Y}(\mathbf{r})) \equiv C^o(\|\mathbf{s} - \mathbf{r}\|) = \left(1 + \frac{\sqrt{3}(\|\mathbf{s} - \mathbf{r}\|)}{\ell}\right) \exp\left(-\frac{\sqrt{3}(\|\mathbf{s} - \mathbf{r}\|)}{\ell}\right), \quad \mathbf{s}, \mathbf{r} \in D,$$

where we set the length scale $\ell = 1$. When calibrating we found it useful to use detrended zero mean realisations of the target process since, without further modification (e.g., exponentiation

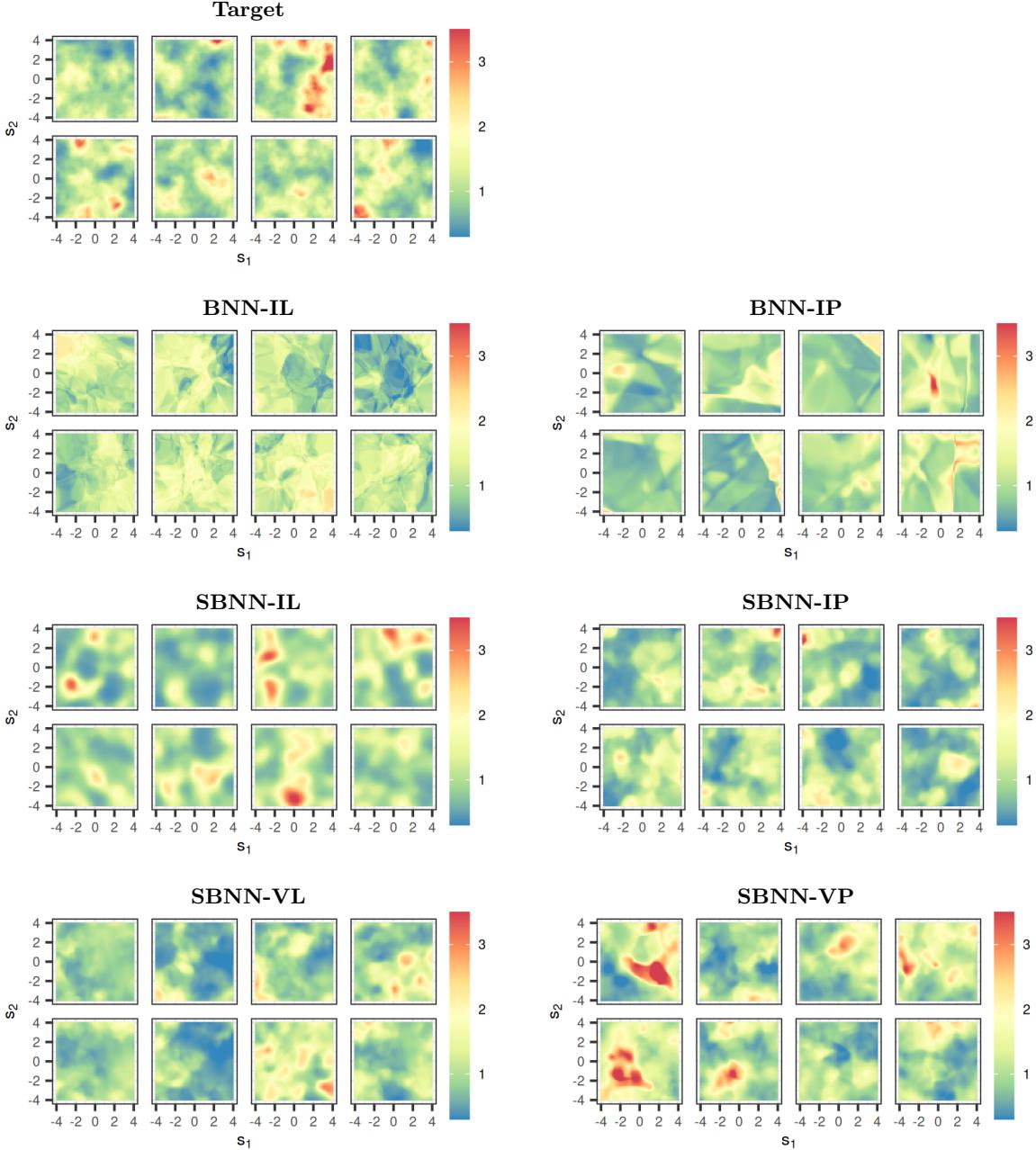


Fig. 8: Samples from the target lognormal process and all six (S)BNN variants calibrated to the target process.

of the output from the neural network), our (S)BNNs are not designed for positive-only processes. After calibration, the true mean of $\tilde{Y}(\cdot)$ was then added to the output of our (S)BNNs to ensure that realisations from the calibrated processes have the correct mean.

We show realisations from the target process $\tilde{Y}(\cdot)$ and the calibrated models in Figure 8. The SBNNs are clearly able to capture the stochastic properties of the target process, with the realisations from the calibrated SBNN-IP and SBNN-VP appearing to be qualitatively more similar to those of the target process than those from the calibrated SBNN-IL and SBNN-VL. Realisations from the calibrated non-spatial BNNs are clearly very different to what one could expect from the target process. This corroborates Table 1, where we see that the ‘prior-per-parameter’ SBNN variants have a relatively low Wasserstein distance $W_1(\cdot)$. Note that $W_1(\cdot)$ for the non-spatial BNN-IP is similar to that for the SBNN-VL, and yet the sample paths from the former model are clearly not similar to those from the target process, indicating the importance

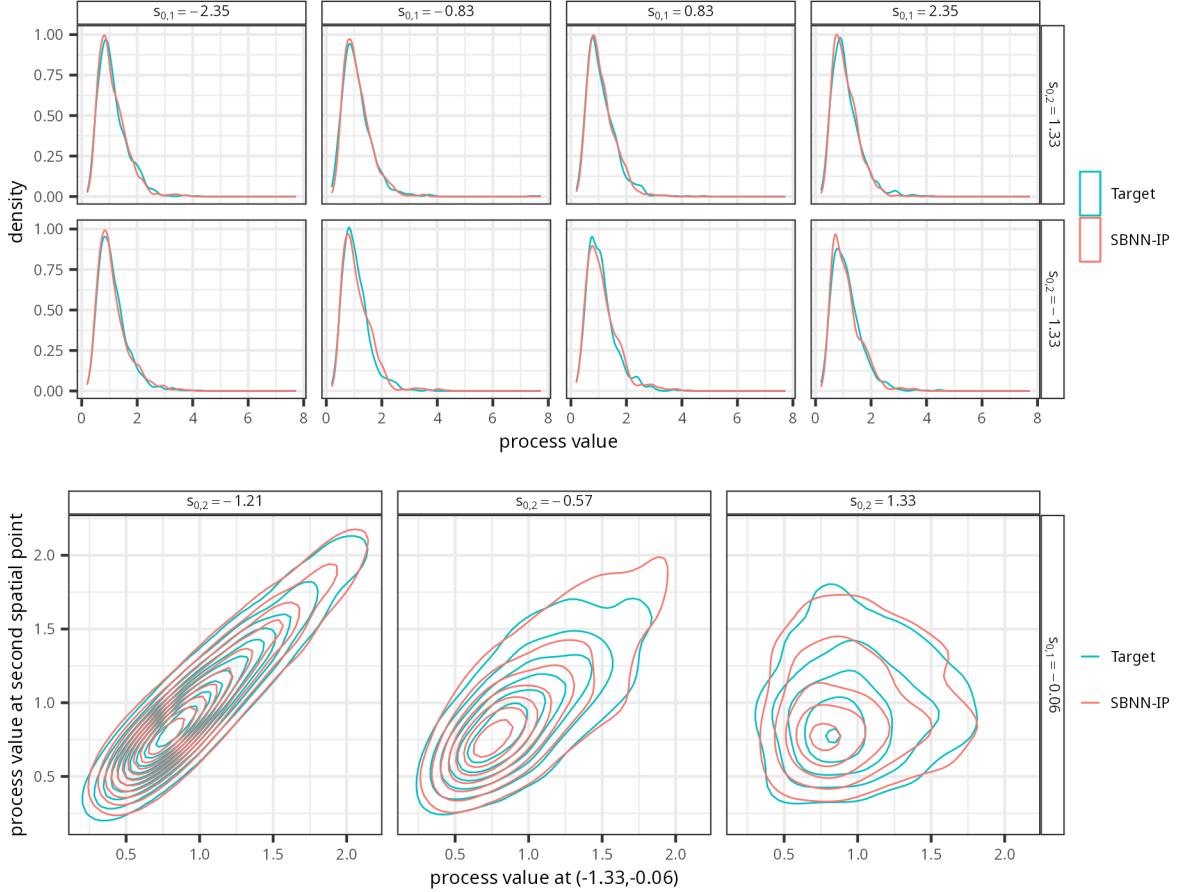


Fig. 9: Kernel density plots from 1000 samples of the SBNN-IP and the target lognormal spatial process. (Top panel) Univariate densities of the two processes at eight spatial points arranged on a 2×4 grid in D (with coordinates as indicated by the labels of the sub-panels). (Bottom panel) Bivariate densities of the two processes at $\tilde{s}_0 = (-1.33, -0.06)'$ and three other points in D (with coordinates as indicated by the labels of the sub-panels). Contour levels (from the outer to the inner contours) correspond to 0.20, 0.40, 0.60, 0.80, 1.00, 1.20, 1.40, 1.60, 1.80, 2.00, 2.20 and 2.40.

of following up the calibration with other diagnostics such as inspection of sample paths.

Figure 9 is analogous to Figure 5 but for the SBNN-IP (the ‘best’ model according to Table 1) and the lognormal process. That is, the figure shows marginal and bivariate kernel density estimates at selected points in D from 1000 samples taken from the calibrated SBNN-IP and the target lognormal process. As in the Gaussian case, both the marginal and the joint densities are similar. Density plots for the SBNN-VL, shown in Figure S7 in the Supplementary Material, show that the SBNN-V with the ‘prior-per-layer’ scheme is well-calibrated, but that it is unable to capture the lower tails well.

Overall, these results suggest that our SBNNs are able to model non-Gaussian processes, and that they potentially could be applied to a much wider class of models than those considered here, notably models whose likelihood function is intractable but which can be simulated from relatively easily (e.g., models of spatial extremes, [Davison and Huser, 2015](#)).

5. Making inference with SBNNs

Once an SBNN has been calibrated, it serves two purposes: (i) to efficiently simulate realisations of an underlying stochastic process; and (ii) to make inference conditional on observational data. Unconditional simulation is straightforward to do once an SBNN is calibrated, by using

one of the model specifications outlined in Section 2.2.3 with the free hyper-parameters, ψ , replaced with the optimised ones, ψ^* . In this respect the SBNN could be used as a surrogate for a computationally-intensive stochastic simulator, such as a stochastic weather generator (Semenov et al., 1998; Kleiber et al., 2023). Inference, on the other hand, requires further substantial computation. Specifically, given a data set $\mathbf{Z} \equiv (Z_1, \dots, Z_m)'$ of noisy measurements of $Y(\cdot)$ collected at locations $\tilde{\mathbf{s}}_1, \dots, \tilde{\mathbf{s}}_m \in D$, inference proceeds by evaluating, approximating, or sampling from, the posterior distribution of the weights and biases; namely, the distribution of the neural network parameters conditional on \mathbf{Z} and the calibrated hyper-parameters ψ^* . After sampling from the posterior distribution of the weights and biases, it is straightforward to obtain samples from the posterior distribution of $Y(\cdot)$, which we refer to as the predictive distribution.

Several inferential methods have been developed for BNNs (e.g., Jospin et al., 2022). These include variational inference (e.g., Zammit-Mangion et al., 2022) and MCMC. Among MCMC techniques, Hamiltonian Monte Carlo (Neal, 1996) is the most widely used. The original HMC algorithm of Neal (1996) was *full-batch*; that is, it used the entire dataset \mathbf{Z} for generating each posterior draw. However, the gradient computations required for HMC become computationally infeasible for large datasets. Chen et al. (2014) proposed to resolve these computational limitations by approximating the gradient at each MCMC iteration using a mini-batch of the data. The resulting gradient approximation is termed a stochastic gradient, and the corresponding approximation to HMC is termed stochastic gradient Hamiltonian Monte Carlo (SGHMC). SGHMC and its adaptive variant (Springenberg et al., 2016) are well-suited for making inference with BNNs.

It is straightforward to apply SGHMC to the SBNN-I variants since one can readily make use of available software for BNNs. For illustration, here we obtain predictive distributions for $Y(\cdot)$ the SBNN-IL calibrated to the stationary Gaussian distribution of Section 4.1. The top two panels of Figure 10 show the prediction (posterior mean) and the prediction standard error (posterior standard deviation) of the calibrated SBNN-IL, when conditioned on data \mathbf{Z} with sample locations denoted by the black dots. Here, the data model for \mathbf{Z} is given by $Z_i = Y(\tilde{\mathbf{s}}_i) + \epsilon_i$, for $i = 1, \dots, m$, where $m = 100$, $\epsilon_i \sim \text{Gau}(0, 0.001)$, and where $\tilde{\mathbf{s}}_i \in D$ are the sample locations. The predictions and the prediction standard errors under the true model are shown in the bottom two panels. Note how these quantities are similar in pattern; this was somewhat expected since, as shown in Section 4.1, the SBNN-IL is well-calibrated to the true underlying process. However, there are also some differences, particularly the scale of the posterior standard deviations. We now explore the differences further. In Figure 11 we plot kernel density estimates from 800 MCMC samples taken from the predictive distribution under the calibrated SBNN-IL and the true (also target) stationary Gaussian spatial process. The top and bottom panels show empirical marginal predictive densities and joint bivariate predictive densities corresponding to the predictive distributions of the processes at the same spatial locations considered in Figure 5. The marginal and joint predictive densities of the SBNN-IL appear to be unimodal and Gaussian; this is reassuring since any finite-dimensional distribution of the target process conditioned on the data \mathbf{Z} is indeed Gaussian. On the other hand, we now also observe some discrepancy in the shapes of these posterior densities, with the SBNN-IL appearing have a smaller peak and heavier tails. We speculate that this discrepancy may be due to the MCMC chains still needing more time to converge: Although we ran four parallel MCMC chains for 300,000 iterations in each chain, excluding 100,000 samples as burn-in, and thinning by a factor of 1,000, our effective sample size ranged between 50–400 at several prediction locations. This points to the difficulty in making accurate spatial prediction with SBNNs, generally, and that more work is needed to facilitate this “second stage” in our paradigm.

Spatial prediction with the calibrated SBNN-VI and the calibrated SBNN-VP is even more onerous, and it would require some alterations to standard algorithms available for BNNs, which are designed to sample from the posterior distribution of the weights and the biases that do

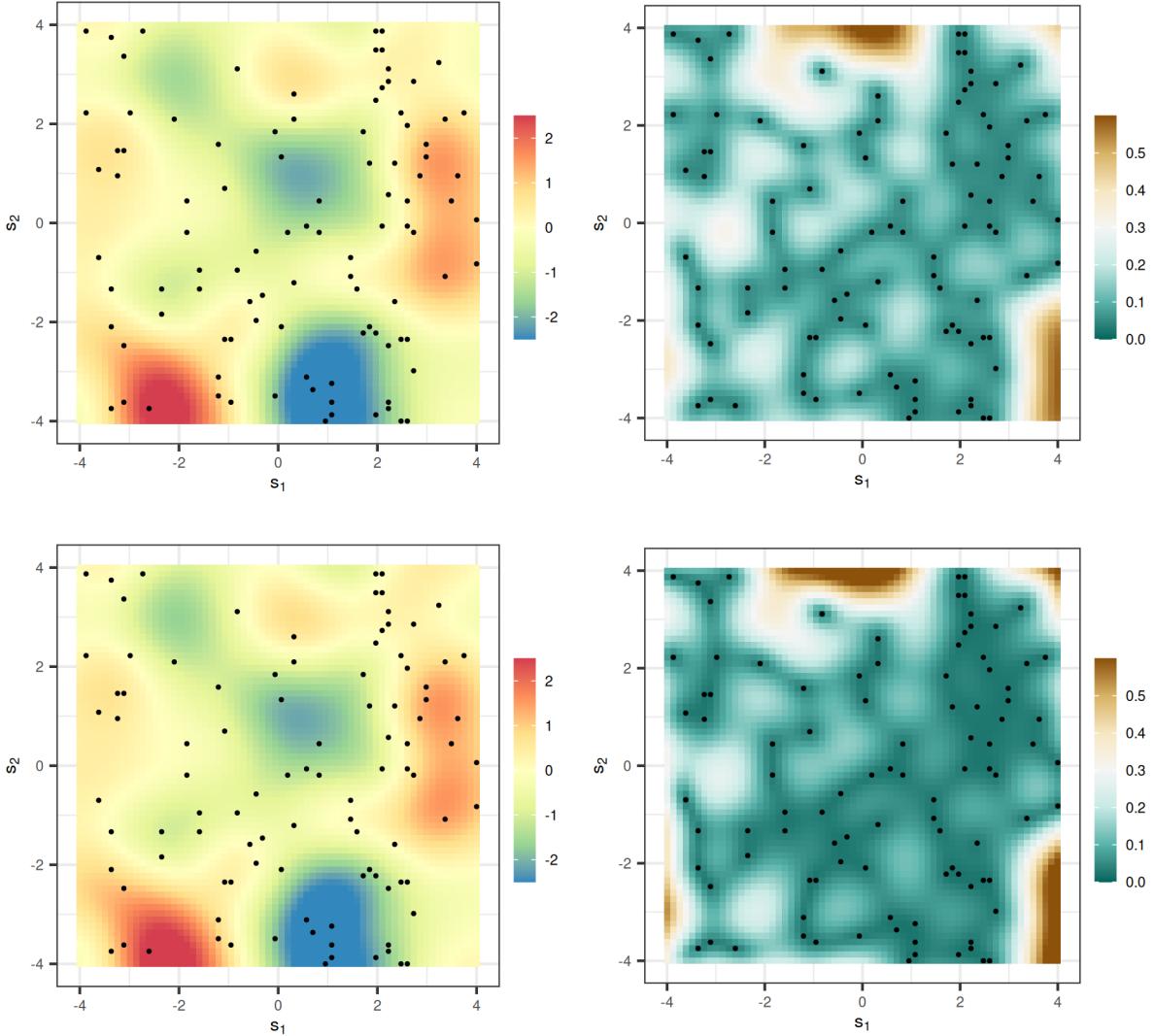


Fig. 10: The predictions (left panels) and prediction standard errors (right panels) of the process evaluated at locations on a fine gridding of D . The data have locations denoted by the black dots. (Top panels) Predictive distribution from the calibrated SBNN-IL, found using stochastic gradient Hamiltonian Monte Carlo. (Bottom panels) The true predictive distribution evaluated from the target process model using multivariate Gaussian distribution formulas.

not vary as a function of the input. Since our SBNN-V formulations in Section 2.2.3 admit a low-dimensional representation of the spatially-varying parameters, in our case we would only need to sample from the conditional distribution of the η -parameters in (9) and (10) given the data. While this would require some changes to standard SGHMC software routines for SBNNs to accommodate this, we do not envision this to be problematic. The development of efficient MCMC algorithms for the SBNN-V variants is the subject of future work.

6. Conclusion

The proposed spatial-statistical methodology that calibrates SBNNs to a target spatial process differs starkly from current approaches where one uses a parametric model as a starting point, estimates the model parameters, and then uses the fitted model to predict at unobserved locations. Instead, with SBNNs, one first calibrates the prior distribution of the weights and

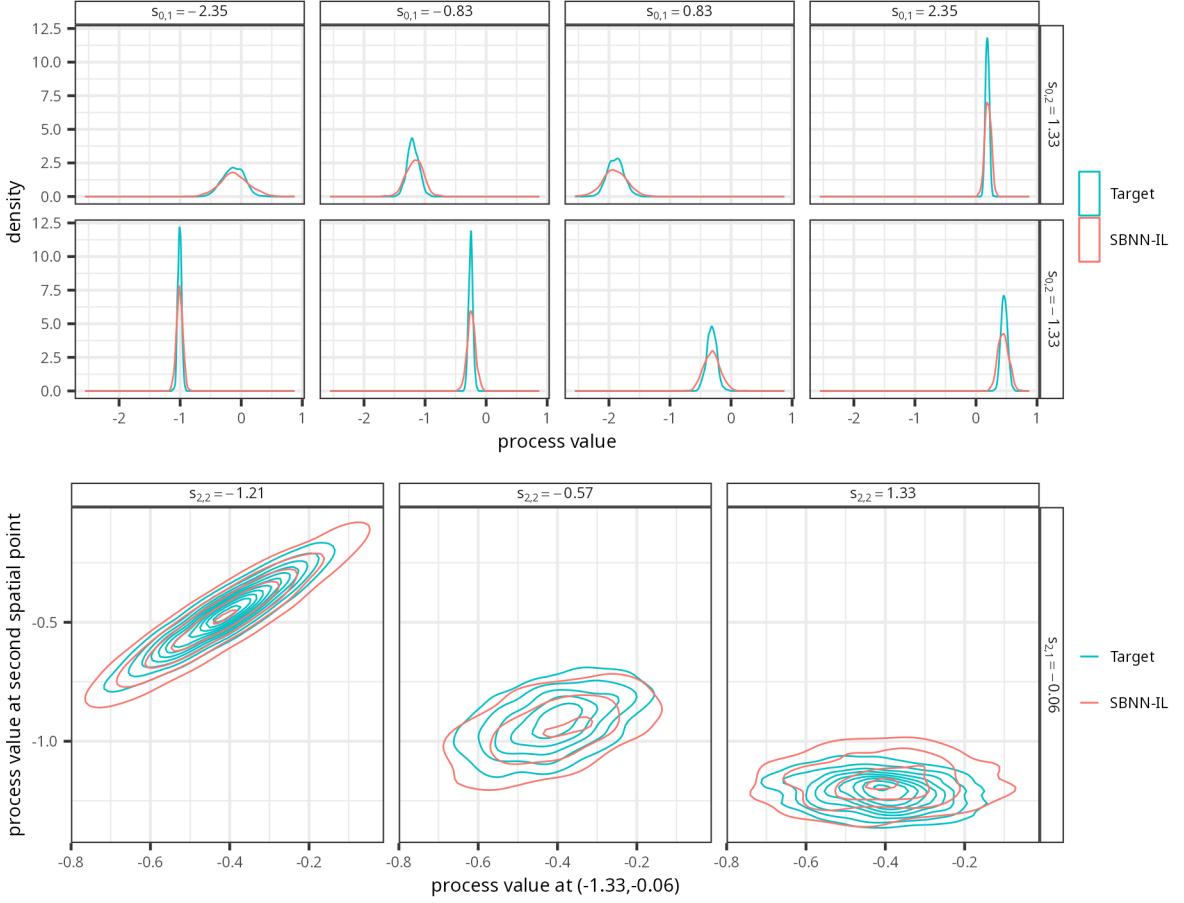


Fig. 11: Kernel density plots from 800 samples from the predictive distribution of the SBNN-IL and that of the target stationary Gaussian spatial process. (Top panel) Marginal predictive densities of the two processes at eight spatial points arranged on a 2×4 grid in D (with coordinates as indicated by the labels of the sub-panels). (Bottom panel) Bivariate predictive densities of the two processes at $\mathbf{s} = (-1.33, -0.06)'$ and three other points in D (with coordinates as indicated by the labels of the sub-panels). Contour levels (from the outer to the inner contours) correspond to 2.00, 4.00, 6.00, 8.00, 10.00, 12.00, 14.00, 16.00, 18.00, 20.00 and 22.00.

biases, then finds the posterior distribution of these weights and biases, and then uses this to obtain predictive distributions of the process. We show that a single SBNN can be used to model a large variety of spatial processes that could be non-stationary and/or non-Gaussian. This concluding section focusses on the advantages and disadvantages of SBNNs, shedding light on situations where they will likely be useful and where they will not.

6.1. Calibration requires replicated realisations from the underlying stochastic process

The biggest limitation of SBNNs, in our view, is that their calibration requires a considerable number of realisations from an underlying stochastic process. In many applications of interest, the spatial statistician only has a single realisation at hand. In such cases, one could calibrate an SBNN to a process model that is easy to simulate from, but it is unclear whether this approach will ultimately lead to any computational or inferential benefit. For example, several methods exist for parameter estimation and prediction with Gaussian processes, and there is little reason why one needs to calibrate SBNNs to Gaussian processes other than for software verification. On the other hand, there are other processes (e.g., max-stable processes and some classes of stochastic partial differential equations), where parameter estimation, prediction, and conditional simulation are notoriously difficult, and yet the process is relatively easy to (unconditionally) simulate from. In such cases, SBNNs are well-positioned to yield computational

benefits over the classical approach of fitting and predicting using a parametric model.

When a large quantity of data are available, and where these data can be reasonably treated as replicates from some underlying stochastic process, SBNNs present a natural way forward to modelling and predicting. They place few assumptions on the underlying process and relieve the modeller from having to make a judgement call on which class of models is most suited to their application. Calibration data could be available, for example, in the form of re-analysis data of some geophysical quantity (such as sea-surface temperature). For example, temporally stationary spatio-temporal data could provide the spatial replicates indexed by time.

6.2. Computing resources

Both calibrating SBNNs and finding the posterior distribution of their parameters requires considerable computing resources and sophisticated algorithms. However, once these routines have been developed, they are widely applicable; this leads to several advantages from a computing perspective. First, since SBNNs are process agnostic, calibrating and fitting them to data is likely to require a similar amount of computing resources, irrespective of what the underlying target process is. This is not the case with classical approaches, where the adopted spatial process model largely determines the computational complexity involved when estimating and predicting. Second, while SBNNs can be used in a wide range of settings, the same algorithms for calibration and for making inference can be used. This is in contrast to traditional likelihood-based approaches where the model is application-dependent with parameter spaces of varying sizes, and where complex algorithms are generally needed for complex models. Finally, with SBNNs, model calibration can be done using gradient descent with mini-batches (i.e., a small set of realisations at each optimisation step); this allows one to calibrate SBNNs using a large number of realisations on memory-limited devices.

6.3. Computational tools used for facilitating calibration

Our approach to calibration substitutes the 1-Lipschitz function, which is required for establishing the Wasserstein distance between two distributions, with a function that is only approximately 1-Lipschitz. The most straightforward way to remove this approximation, and to ensure 1-Lipschitz continuity of $\phi_{NN}(\cdot; \boldsymbol{\lambda})$, is to construct a neural network $\phi_{NN}(\cdot; \boldsymbol{\lambda})$ that is 1-Lipschitz by design. For instance, as 1-Lipschitz functions are closed under composition, a 1-Lipschitz multi-layer perceptron can be constructed by composing 1-Lipschitz affine transformations with 1-Lipschitz activations that are themselves 1-Lipschitz (e.g., hyperbolic tangent functions, ReLUs, and sigmoid functions). Affine transformations can be made 1-Lipschitz by enforcing norm constraints on the weight matrices (Anil et al., 2019). This can be implemented using, for example, spectral normalisation, which re-scales the weight matrices using their dominant singular values, or Björk orthonormalisation, which projects the weight matrices onto their nearest orthonormal neighbours (Ducotterd et al., 2022). These alternative approaches were tried and led to computational difficulties in our setting, and thus were put aside in favour of the more computationally-efficient approach of Gulrajani et al. (2017).

Even if we were successful in establishing the exact Wasserstein distance between two distributions, our approach only aims at matching one selected finite-dimensional distribution of the SBNN to that of the target process. However, when generating samples for (12), one may also randomly sample the spatial locations (and the number thereof) over which both $Y(\cdot)$ and $\tilde{Y}(\cdot)$ are evaluated. One would then be targeting a wide range of finite-dimensional distributions using the SBNN, and not just one; the optimised hyper-parameters ψ^* would then lead to an SBNN $Y(\cdot)$ that approximates the entire process $\tilde{Y}(\cdot)$ in distribution. Note that even if all the finite-dimensional distributions of the two processes are identical, their sample paths may differ; see Lindgren (2012, Section 2.2) for discussion and examples. This caveat is unlikely to cause issues in practice unless the sample paths of $Y(\cdot)$ or $\tilde{Y}(\cdot)$ are discontinuous.

6.4. SBNN architecture and model interpretation

Incorporating an embedding layer and spatially-varying network parameters was needed for our SBNNs to be able to reproduce realistic covariances and covariance non-stationarity/anisotropy. We have yet to explore what spatial processes our SBNN architectures are not suitable for. Future work may reveal that our SBNNs are too inflexible in some settings, even with a ‘prior-per-parameter’ scheme, and that further modifications are needed. A drawback of our SBNNs is that they are by-and-large uninterpretable: Whereas in classical modelling one obtains estimates or distributions over parameters that typically have a clear interpretation, with an SBNN one only has access to posterior distributions over weights and biases that are related to the output in a highly nonlinear fashion. This limitation alone may preclude SBNNs from being used in certain settings where parameter interpretability is paramount. On the other hand, where prediction and uncertainty quantification through predictive variances is the main goal, SBNNs are likely to be of high practical value.

7. Acknowledgements

AZ-M was partially supported the Australian Research Council (ARC) Discovery Early Career Research Award DE180100203. MF gratefully acknowledges support from the AXA Research Fund. Further, this material is based upon work supported by the Air Force Office of Scientific Research under award number FA2386-23-1-4100 (AZ-M and NC).

References

- Anil, C., Lucas, J., Grosse, R., 2019. Sorting out Lipschitz function approximation, in: Chaudhuri, K., Salakhutdinov, R. (Eds.), Proceedings of the 36th International Conference on Machine Learning, PMLR, Long Beach, CA. pp. 291–301.
- Buhmann, M.D., 2003. Radial Basis Functions: Theory and Implementations. Cambridge University Press, Cambridge, UK.
- Chan, J., Perrone, V., Spence, J., Jenkins, P., Mathieson, S., Song, Y., 2018. A likelihood-free inference framework for population genetic data using exchangeable neural networks, in: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (Eds.), Advances in Neural Information Processing Systems, Curran Associates, Inc., Red Hook, NY. pp. 8594–8605.
- Chen, T., Fox, E., Guestrin, C., 2014. Stochastic gradient Hamiltonian Monte Carlo, in: Xing, E.P., Jebara, T. (Eds.), Proceedings of the 31st International Conference on Machine Learning, PMLR, Bejing, China. pp. 1683–1691.
- Chen, W., Li, Y., Reich, B.J., Sun, Y., 2023. DeepKriging: Spatially dependent deep neural networks for spatial prediction. *Statistica Sinica* , doi: 10.5705/ss.202021.0277 (in press).
- Cressie, N., 1993. Statistics for Spatial Data, revised edition. John Wiley & Sons, New York, NY.
- Cressie, N., Johannesson, G., 2008. Fixed rank kriging for very large spatial data sets. *Journal of the Royal Statistical Society B* 70, 209–226.
- Davison, A.C., Huser, R., 2015. Statistics of extremes. *Annual Review of Statistics and its Application* 2, 203–235.
- De Oliveira, V., Kedem, B., Short, D.A., 1997. Bayesian prediction of transformed Gaussian random fields. *Journal of the American Statistical Association* 92, 1422–1433.

- Delattre, S., Fournier, N., 2017. On the Kozachenko-Leonenko entropy estimator. *Journal of Statistical Planning and Inference* 185, 69–93.
- Ducotterd, S., Goujon, A., Bohra, P., Perdios, D., Neumayer, S., Unser, M., 2022. Improving Lipschitz-constrained neural networks by learning activation functions. Online: <https://arxiv.org/abs/2210.16222>.
- Dunlop, M.M., Girolami, M.A., Stuart, A.M., Teckentrup, A.L., 2018. How deep are deep Gaussian processes? *Journal of Machine Learning Research* 19, 1–46.
- Duvenaud, D., Rippel, O., Adams, R., Ghahramani, Z., 2014. Avoiding pathologies in very deep networks, in: Kaski, S., Corander, J. (Eds.), *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, PMLR, Reykjavik, Iceland. pp. 202–210.
- Flam-Shepherd, D., Requeima, J., Duvenaud, D., 2017. Mapping Gaussian process priors to Bayesian neural networks. Presented at the NeurIPS workshop on Bayesian Deep Learning. Online: <http://bayesiandeeplearning.org/2017/papers/65.pdf>.
- Goodfellow, I., Bengio, Y., Courville, A., 2016. *Deep Learning*. MIT press, Cambridge, MA.
- Graves, A., 2011. Practical variational inference for neural networks, in: Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., Weinberger, K. (Eds.), *Advances in Neural Information Processing Systems 24*, Curran Associates Inc., Red Hook, NY. pp. 2348–2356.
- Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A., 2017. Improved training of Wasserstein GANs, in: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.), *Advances in Neural Information Processing Systems 31*, Curran Associates, Inc., Red Hook, NY. pp. 5768–5778.
- He, K., Zhang, X., Ren, S., Sun, J., 2015. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification, in: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1026–1034. doi:10.1109/ICCV.2015.8.
- He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778. doi:10.1109/CVPR.2016.90.
- Jacot, A., Gabriel, F., Hongler, C., 2018. Neural tangent kernel: Convergence and generalisation in neural networks, in: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (Eds.), *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., Red Hook, NY. pp. 8580–8569.
- Jospin, L.V., Laga, H., Boussaid, F., Buntine, W., Bennamoun, M., 2022. Hands-on Bayesian neural networks – a tutorial for deep learning users. *IEEE Computational Intelligence Magazine* 17, 29–48.
- Kirkwood, C., Economou, T., Pugeault, N., Odber, H., 2022. Bayesian deep learning for spatial interpolation in the presence of auxiliary information. *Mathematical Geosciences* 54, 507–531.
- Kleiber, W., Sain, S., Madaus, L., Harr, P., 2023. Stochastic tropical cyclone precipitation field generation. *Environmetrics* 34, e2766.
- Kochenderfer, M.J., Wheeler, T.A., 2019. *Algorithms for Optimization*. MIT Press, Cambridge, MA.
- Lindgren, G., 2012. *Stationary Stochastic Processes: Theory and Applications*. CRC Press, Boca Raton, FL.

- MacKay, D.J., 1992. A practical Bayesian framework for backpropagation networks. *Neural Computation* 4, 448–472.
- Malinin, A., Chervontsev, S., Prosvirkov, I., Gales, M., 2020. Regression prior networks. Online: <https://arxiv.org/abs/2006.11590>.
- McDermott, P.L., Wikle, C.K., 2019. Bayesian recurrent neural network models for forecasting and quantifying uncertainty in spatial-temporal data. *Entropy* 21, 184.
- Neal, R.M., 1996. Bayesian Learning for Neural Networks. Springer Science & Business Media, New York, NY.
- Nychka, D., Bandyopadhyay, S., Hammerling, D., Lindgren, F., Sain, S., 2018. A multiresolution Gaussian process model for the analysis of large spatial datasets. *Journal of Computational and Graphical Statistics* 24, 579–599.
- Paciorek, C.J., Schervish, M.J., 2006. Spatial modelling using a new class of nonstationary covariance functions. *Environmetrics* 17, 483–506.
- Panaretos, V.M., Zemel, Y., 2019. Statistical aspects of Wasserstein distances. *Annual Review of Statistics and its Application* 6, 405–431.
- Payares-Garcia, D., Mateu, J., Schick, W., 2023. Spatially informed Bayesian neural network for neurodegenerative diseases classification. *Statistics in Medicine* 42, 105–121.
- Rasmussen, C.E., Williams, K.I., 2006. Gaussian Processes for Machine Learning. MIT Press, Cambridge, MA.
- Semenov, M.A., Brooks, R.J., Barrow, E.M., Richardson, C.W., 1998. Comparison of the WGEN and LARS-WG stochastic weather generators for diverse climates. *Climate Research* 10, 95–107.
- Springenberg, J.T., Klein, A., Falkner, S., Hutter, F., 2016. Bayesian optimization with robust Bayesian neural networks, in: Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., Garnett, R. (Eds.), *Advances in Neural Information Processing Systems* 29, Curran Associates, Inc., Red Hook, NY. pp. 4141–4149.
- Tran, B.H., Rossi, S., Milios, D., Filippone, M., 2022. All you need is a good functional prior for Bayesian deep learning. *Journal of Machine Learning Research* 23, 1–56.
- Zammit-Mangion, A., Cressie, N., 2021. FRK: An R package for spatial and spatio-temporal prediction with large datasets. *Journal of Statistical Software* 98, 1–48.
- Zammit-Mangion, A., Ng, T.L.J., Vu, Q., Filippone, M., 2022. Deep compositional spatial models. *Journal of the American Statistical Association* 117, 1787–1808.

Supplementary Material for “Spatial Bayesian Neural Networks”

Andrew Zammit-Mangion, Michael D. Kaminski, Ba-Hien Tran, Maurizio Filippone, Noel Cressie

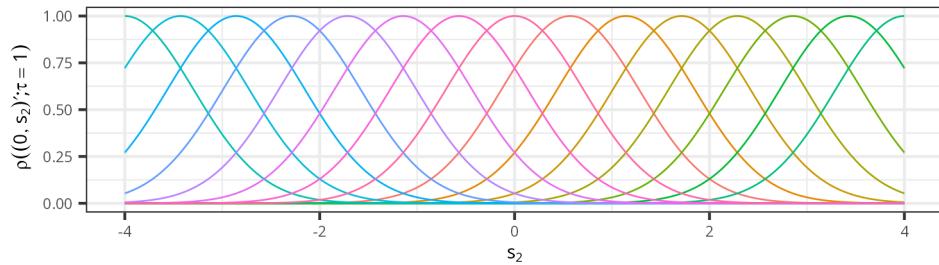


Fig. S1: Evaluations of the basis functions in $\rho(\cdot; \tau = 1)$, which have their centroid at $s_1 = 0$, along an s_2 -transect at $s_1 = 0$. The different colours denote the different basis functions, of which there are $\sqrt{K} = 15$.

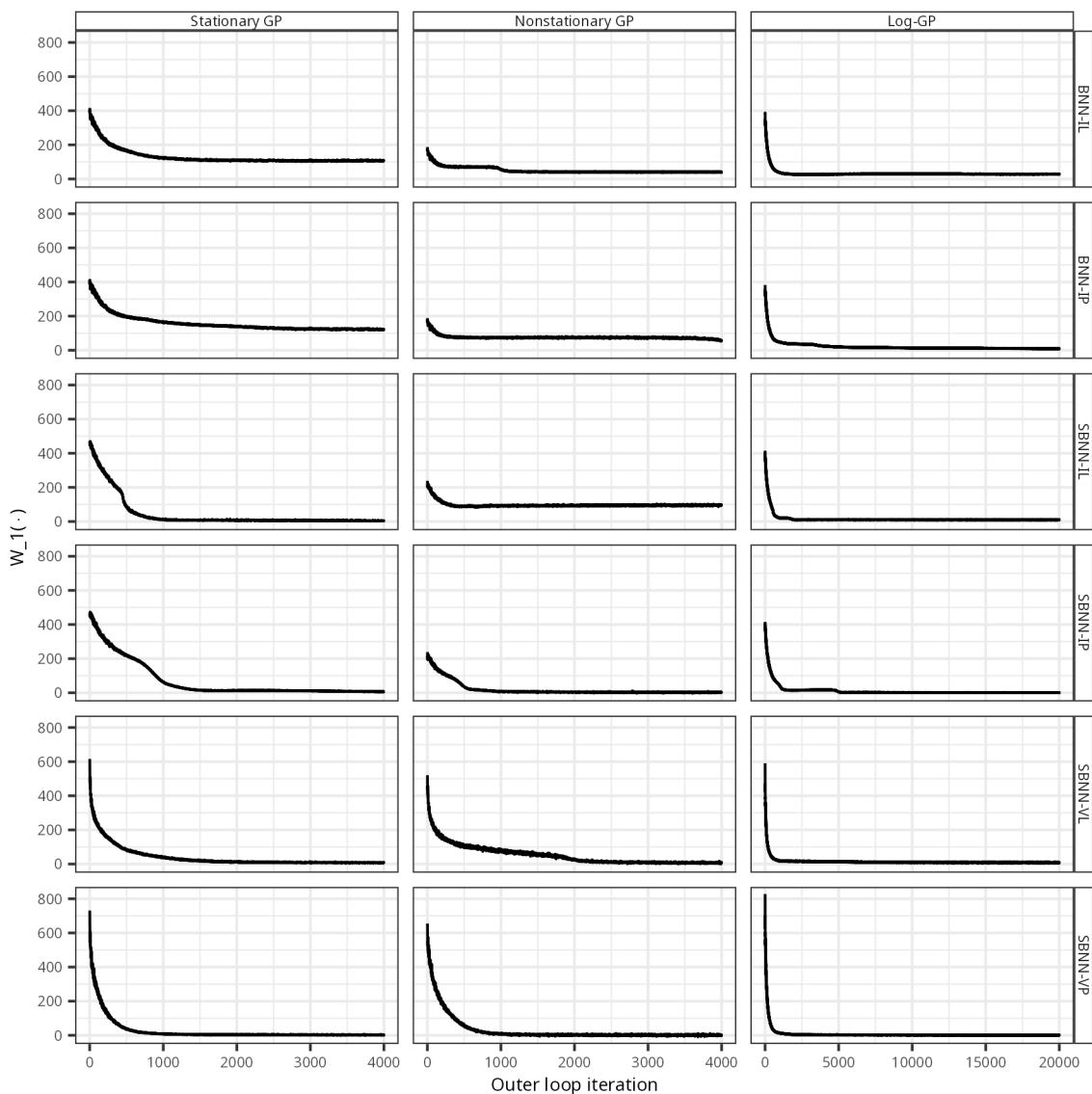


Fig. S2: Wasserstein distances $W_1(\cdot)$ as a function of outer-loop iteration number when training the six models for the three simulation experiments

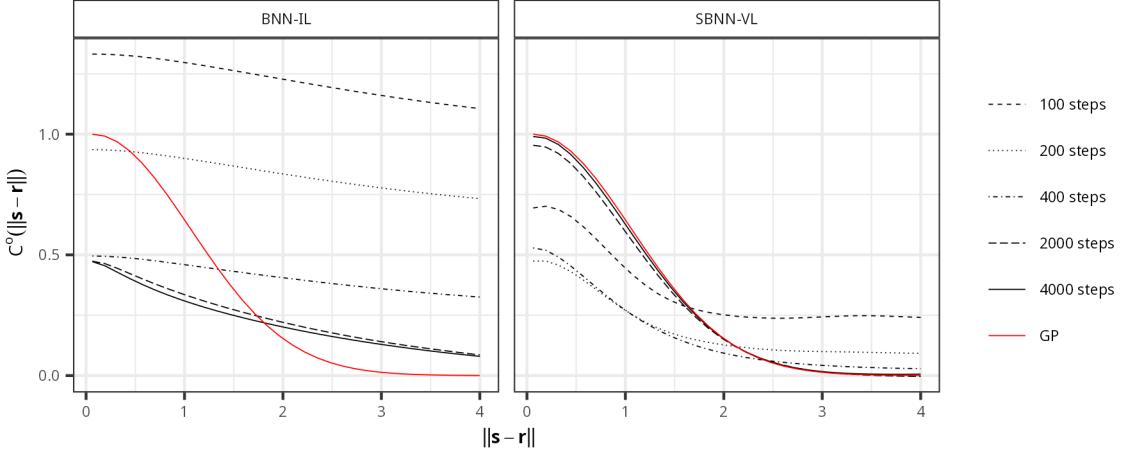


Fig. S3: Empirical covariogram of the (S)BNN at different stages during the optimisation (different line styles denote the empirical covariogram after 100, 200, 400, 1000, 2000, and 4000 gradient steps, respectively), and the true covariogram of the target Gaussian process (red). Left: BNN-IL. Right: SBNN-VL.

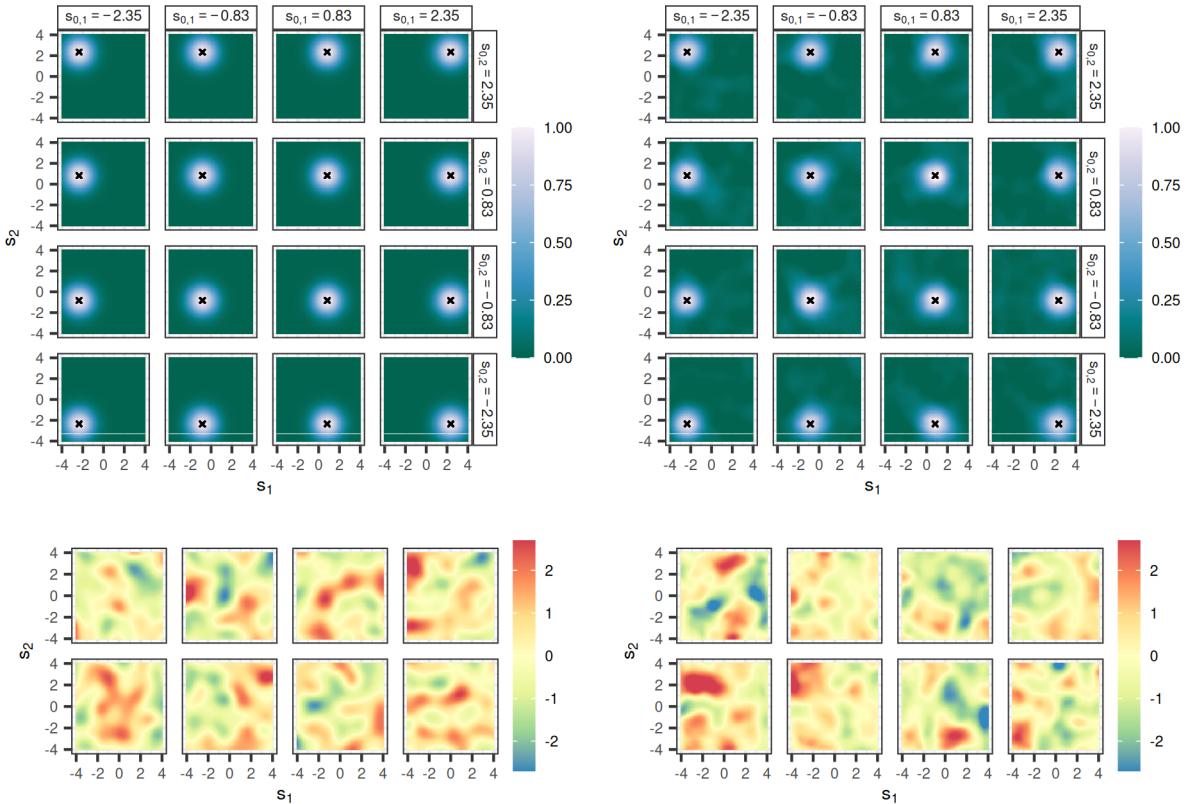


Fig. S4: (Top-left panel) Covariance between the target process at 16 grid points (crosses) with coordinates as indicated by the labels at the top and the right of the sub-panels, and the target process on a fine gridding of D . (Bottom-left panel) Eight realisations of the underlying target process on a fine gridding of D . (All right panels) Same as left panels but for the calibrated SBNN-VL.

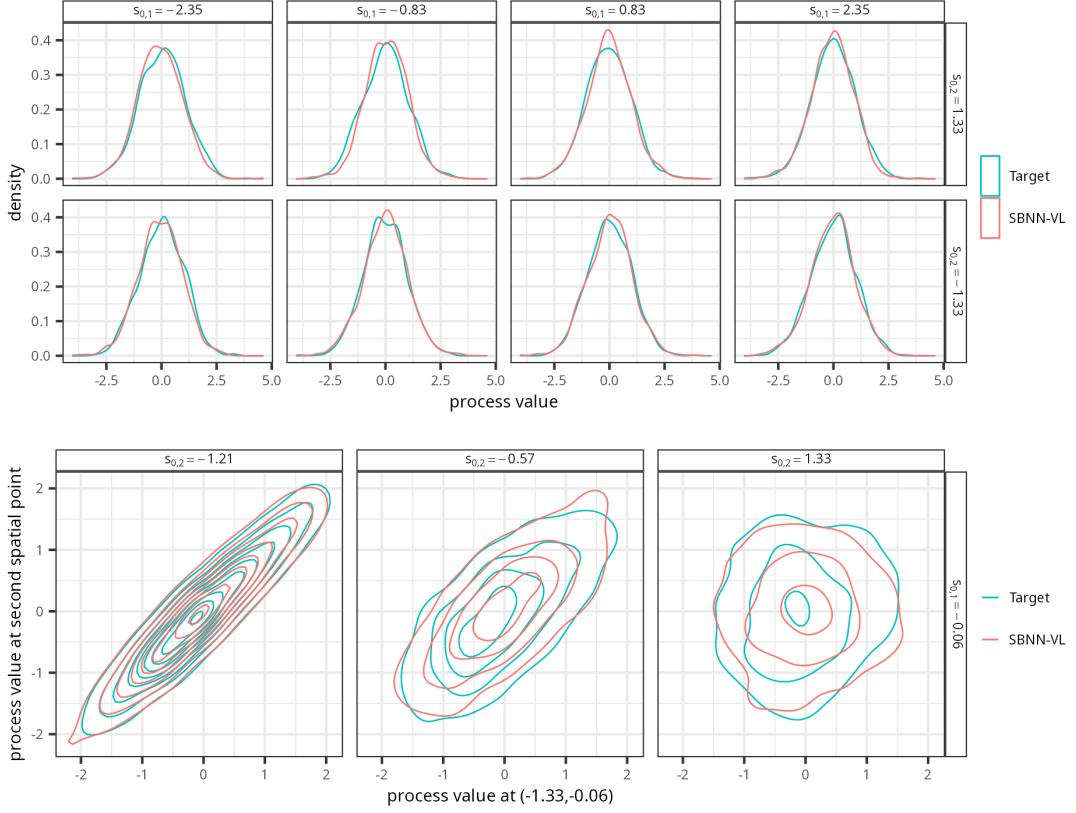


Fig. S5: Kernel density plots from 1000 samples of the SBNN-VL and the target stationary Gaussian spatial process. (Top panel) Univariate densities of the two processes at eight spatial points arranged on a 2×4 grid in D (with coordinates as indicated by the labels of the sub-panels). (Bottom panel) Bivariate densities of the two processes at $\tilde{s}_0 = (-1.33, -0.06)'$ and three other points in D (with coordinates as indicated by the labels of the sub-panels). Contour levels (from the outer to the inner contours) correspond to 0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40 and 0.45.

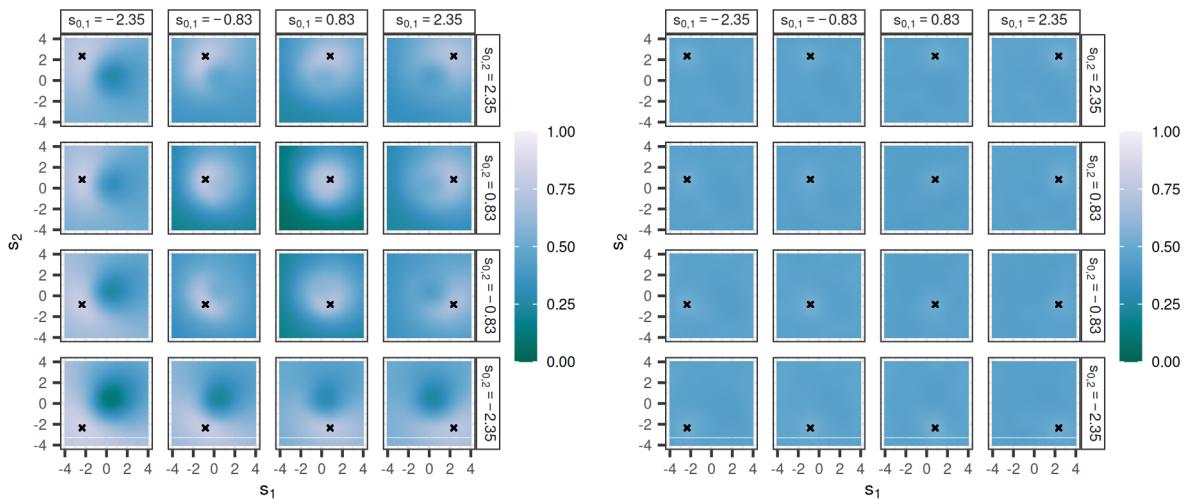


Fig. S6: (Left panel) Covariance between the calibrated BNN-IL process at 16 grid points (crosses) with coordinates as indicated by the labels at the top and the right of the sub-panels, and the process on a fine gridding of D . (Right panel) Same as left panel, but for the calibrated SBNN-IL.

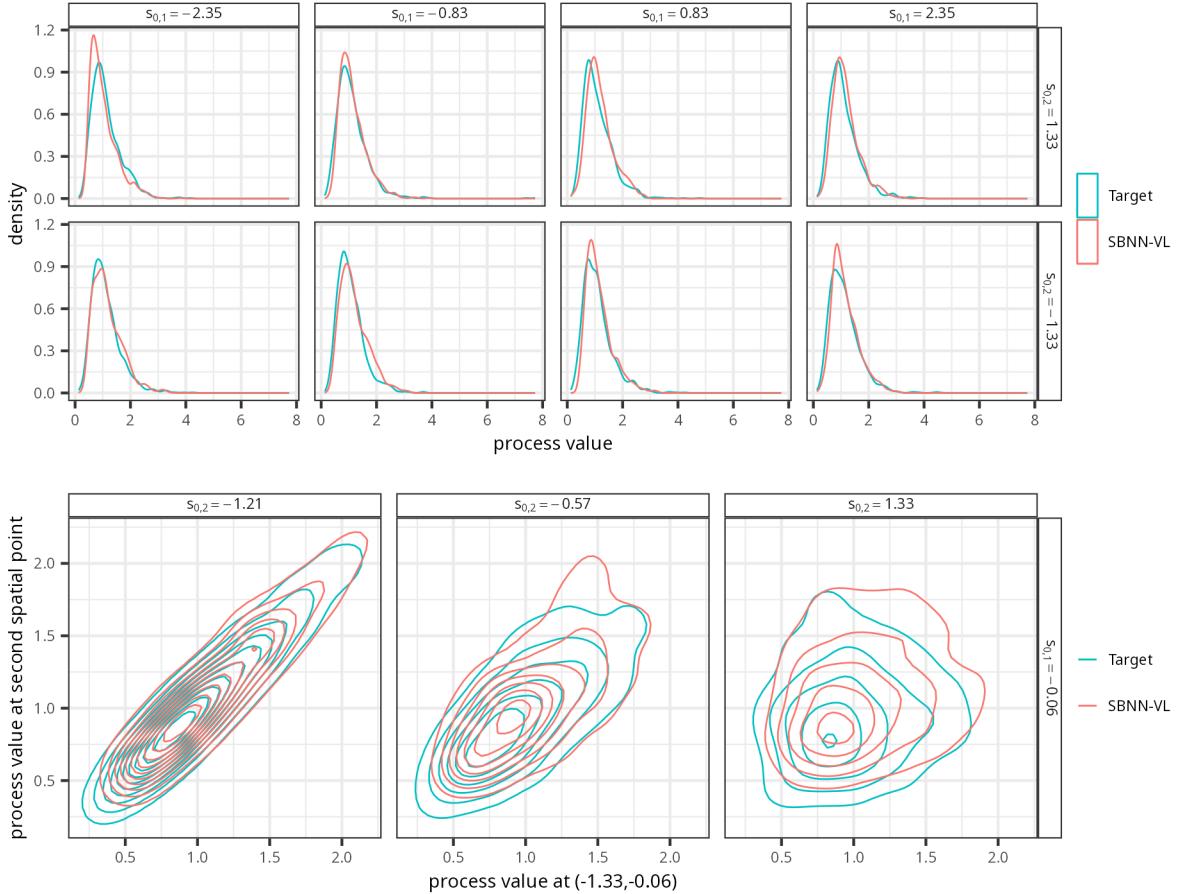


Fig. S7: Kernel density plots from 1000 samples of the SBNN-VL and the target lognormal spatial process. (Top panel) Univariate densities of the two processes at eight spatial points arranged on a 2×4 grid in D (with coordinates as indicated by the labels of the sub-panels). (Bottom panel) Bivariate densities of the two processes at $\tilde{s}_0 = (-1.33, -0.06)'$ and three other points in D (with coordinates as indicated by the labels of the sub-panels). Contour levels (from the outer to the inner contours) correspond to 0.20, 0.40, 0.60, 0.80, 1.00, 1.20, 1.40, 1.60, 1.80 and 2.00.