# DeepRV: Accelerating Spatiotemporal Inference with Pre-trained Neural Priors

**Jhonathan Navott**[1,2*] **Daniel Jenson**[2,*] **Seth Flaxman**[2] **Elizaveta Semenova**[1,†]
[1]School of Public Health, Imperial College London, UK
[2]Department of Computer Science, University of Oxford, UK

## Abstract

Gaussian Processes (GPs) provide a flexible and statistically principled foundation for modelling spatiotemporal phenomena, but their $\mathcal{O}(N^3)$ scaling makes them intractable for large datasets. Approximate methods such as variational inference (VI), inducing-point (sparse) GPs, low-rank kernel approximations (*e.g.*, Nyström methods and random Fourier features), and approximations such as INLA improve scalability but typically trade off accuracy, calibration, or modelling flexibility. We introduce **DeepRV**, a neural-network surrogate that *replaces GP prior sampling*, while closely matching full GP accuracy at inference including hyperparameter estimates, and reducing computational complexity to $\mathcal{O}(N^2)$, increasing scalability and inference speed. DeepRV serves as a drop-in replacement for GP prior realisations in *e.g.* MCMC-based probabilistic programming pipelines, preserving full model flexibility. Across simulated benchmarks, non-separable spatiotemporal GPs, and a real-world application to education deprivation in London (n = 4,994 locations), DeepRV achieves the highest fidelity to exact GPs while substantially accelerating inference. Code is provided in the `dl4bi` Python package, with all experiments run on a single consumer-grade GPU to ensure accessibility for practitioners.

*Equal contribution. †Corresponding author: `elizaveta.p.semenova@gmail.com`.

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300. 

## 1 INTRODUCTION

GPs provide a principled Bayesian framework for modelling spatial and spatiotemporal phenomena, offering both predictive accuracy and uncertainty quantification. Their nonparametric nature allows GPs to flexibly capture complex nonlinear relationships without strong assumptions about functional form, while kernel design encodes spatial correlations and domain knowledge. These strengths have driven adoption in disease mapping [Diggle et al., 1998, Diggle and Giorgi, 2015, Zhou and Ji, 2020, Diggle et al., 2013, Lawson, 2018], air pollution modelling [Desai et al., 2022, Patel et al., 2022, Wang et al., 2021, Cheng et al., 2014, Stoddart et al., 2023, Sonabend et al., 2024], and climate risk analysis [Mansour et al., 2024, Agou et al., 2022, Klockmann et al., 2024, Xiong et al., 2021, Wang et al., 2024, Koh et al., 2021]. Importantly, GPs yield interpretable posteriors that enhance decision-making under uncertainty.

As datasets grow, the $\mathcal{O}(N^3)$ cost of GPs renders them computationally infeasible. Approximations such as inducing points [Csató and Opper, 2002, Snelson and Ghahramani, 2006, Quiñonero-Candela and Rasmussen, 2005, Titsias, 2009], low-rank factorizations *e.g.* random Fourier features (RFFs) [Rahimi and Recht, 2007a], variational inference (VI) [Hensman et al., 2013, 2015, Matthews et al., 2017], and the Integrated Nested Laplace Approximation (INLA) [Rue et al., 2009, 2017] enable more scalability, but each trades accuracy for efficiency or imposes restrictive modelling assumptions.

Neural surrogates such as PriorVAE [Semenova et al., 2022], PriorCVAE [Semenova et al., 2023], and $\pi$VAE [Mishra et al., 2022] offer an alternative path, replacing the GP prior with a learned generative decoder to balance flexibility and scalability. These models avoid the cubic cost of exact GP inference, with complexity determined by the decoder architecture (typically quadratic in the number of locations), but often sacrifice accuracy. **DeepRV** provides an alternative and el-

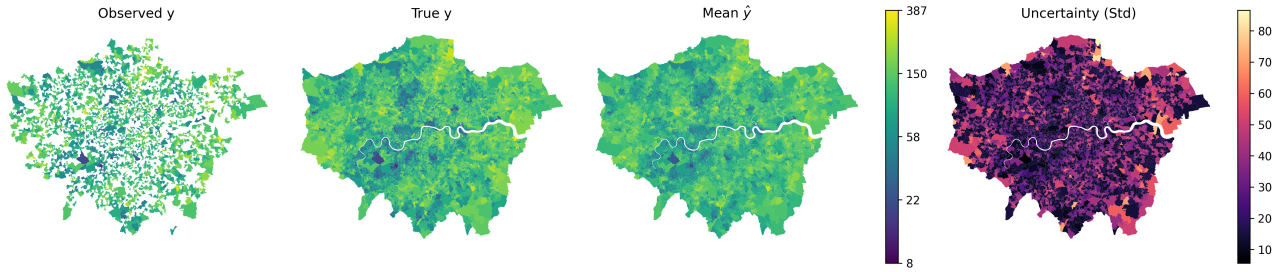


Figure 1: DeepRV predictive evaluation on the London LSOA education deprivation dataset (= 4,994 locations). Panels show (from left to right): observed $\mathbf{y}$ (masked), full true $\mathbf{y}$, DeepRV posterior predictive mean $\hat{\mathbf{y}}$, and DeepRV posterior predictive uncertainty (standard deviation).

egant neural surrogate approach with very high fidelity to full GP inference while substantially improving scalability and speed. We summarise our contributions as follows:

1. The novel **DeepRV** training paradigm for emulating GPs.
2. A comprehensive set of experiments benchmarking INLA, PriorCVAE, RFFs, ADVI, inducing points and DeepRV on 2D Gaussian processes. DeepRV achieves the highest fidelity to full GP Markov Chain Monte Carlo (MCMC) across predictive and parameter metrics, while accelerating MCMC inference substantially.
3. Applying DeepRV to non-separable spatiotemporal GPs, where it flexibly handles covariance structures challenging for INLA and RFFs.
4. Evaluating DeepRV on the education dimension of deprivation in London at the LSOA level (n = 4,994 locations), where standard GP approaches are computationally prohibitive.

Below we review background and related work, introduce DeepRV, and evaluate it across a range of benchmarks and a real-life dataset.

## 2 BACKGROUND

### 2.1 Gaussian Processes (GPs)

A GP is an infinite collection of random variables, any finite subset of which has a joint multivariate Gaussian distribution [Williams and Rasmussen, 2006]. Formally, a stochastic process $\{f(x) : x \in \mathcal{X}\}$ is a GP if for any finite set of inputs, $x_1, \ldots, x_n \in \mathcal{X}$, the random vector

$$\mathbf{f} = (f(x_1), \ldots, f(x_n))^\intercal \tag{1}$$

is distributed as:

$$\mathbf{f} \sim \mathcal{N}(\mu(\mathbf{x}), K(\mathbf{x}, \mathbf{x}')) \tag{2}$$

where $\mu(\mathbf{x}) = [\mu(x_1), \ldots, \mu(x_n)]^\intercal$ is the mean function and $K(\mathbf{x}, \mathbf{x}')$ is the covariance matrix with entries $K_{ij} = k_\theta(x_i, x_j)$, defined by a positive semidefinite kernel function $k_\theta : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ parametrised by $\theta$, which is often the tuple of lengthscale and variance $\theta = (\ell, \sigma^2)$. Thus, a GP can be written as

$$f(x) \sim \mathcal{GP}(\mu(x), k_\theta(x, x')) \tag{3}$$

The kernel function $k_\theta(x, x')$ plays a central role in controlling the smoothness, periodicity, and other structural properties of the functions drawn from the GP prior. Commonly used kernels include the squared exponential / radial basis function, Matérn family, periodic kernel, and linear / polynomial kernel. Combining these kernels through addition and multiplication allow practitioners to model highly structured signals.

### 2.2 GPs for Spatiotemporal Inference

GPs have become a central tool for spatial and spatiotemporal inference, providing a flexible probabilistic framework. In a Bayesian formulation, a GP prior models latent functions such as disease risk, pollution concentration, or meteorological variables over geographical and temporal domains. By defining a covariance structure that encodes correlation, typically as a function of distance, GPs enable coherent interpolation from sparse and irregularly spaced observations to unobserved locations, a task often referred to as kriging [Diggle et al., 1998]. The posterior predictive distribution not only yields point estimates but also quantifies uncertainty, making GPs especially valuable for risk-sensitive applications. Their capacity to integrate prior knowledge through kernel design allows GPs to capture domain-specific structure, while approximate inference methods extend their applicability to increasingly large spatial and spatiotemporal

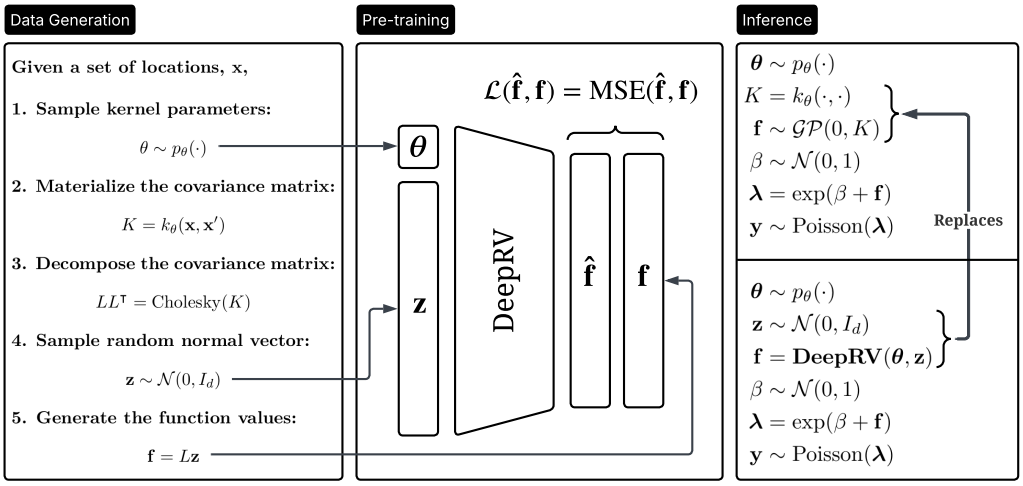


Figure 2: A schematic overview of the DeepRV approach. Left panel details the data generating process used for pre-training. The middle panel shows the input and output of DeepRV during pre-training. In the right panel are two statistical models for inference, the first representing a traditional model that uses a GP prior, while the second one replaces the GP with a trained DeepRV network. In practice, for a given spatiotemporal setting, data generation and pre-training are a one-off cost which takes a few minutes to hours. Once complete, inference runtime is substantially sped-up, and can be repeated as many times as necessary.

datasets. In the subsequent section we review a variety of techniques used to scale GPs for spatial and spatiotemporal inference.

## 3 RELATED WORK

In this section we detail the leading techniques used to scale GPs for inference tasks. We provide a qualitative comparison in Table 1. Further details on these methods, including their specific implementations used in this paper, are provided in Appendix A.

### 3.1 INLA

Integrated Nested Laplace Approximation (INLA) provides a deterministic alternative to MCMC for latent Gaussian models, whose computational cost can be prohibitive for high-dimensional structured settings [Rue et al., 2009]. INLA approximates posterior marginals via nested Laplace approximations coupled with deterministic numerical integration. By exploiting the sparse precision matrices of Gaussian Markov random fields (GMRFs), it enables scalable inference for hierarchical latent Gaussian models widely used in spatial statistics, disease mapping, and environmental risk assessment [Bakka et al., 2018]. The stochastic partial differential equation (SPDE) formulation provides an explicit link between continuously indexed Gaussian fields and discrete GMRFs, facilitating large-scale spatial and spatiotemporal modelling [Lindgren et al., 2011]. In practice, `R-INLA` [Rue et al., 2017] is the primary implementation, and the `inlabru` [Bachl et al., 2019] package builds on it to support richer model specifications, including non-linear predictors via iterative linearisation. Despite these advantages, practical limitations remain: inference is primarily provided as marginal posterior summaries rather than full joint posteriors [Gómez-Rubio and Palmí-Perales, 2017]; the software supports a broad but finite catalogue of likelihoods and latent components, with additional families or extensions requiring non-trivial implementation effort [Rue et al., 2023]; and genuinely non-separable space-time structures need specialised model formulations beyond default workflows [Bakka et al., 2018]. The comparisons performed in this paper rely on the `R-INLA` interface.

### 3.2 Sparse GPs

Sparse GPs introduce a small set of inducing points, $M \ll N$, that are intended to summarize the full dataset, reducing complexity from $\mathcal{O}(N^3)$ to $\mathcal{O}(NM^2)$. Early formulations include pseudo-input GPs [Snelson and Ghahramani, 2006] and the unifying framework of Quiñonero-Candela and Rasmussen [2005], which approximate the covariance structure directly. Titsias [2009] provide a Bayesian framework for learning inducing variables and minimizing information loss, while Hensman et al. [2013] provide a stochastic variational inference extension that enables training on massive datasets using batch optimisation. These methods offer scalability, but often sacrifice accuracy.

Table 1: Qualitative comparison of spatial inference techniques. Citations for each method are in the text.

| Method | Accuracy | Flexibility | Scalability |
|---|---|---|---|
| **GP** | High | High | Low |
| **INLA** | Med-High | Low–Med | High |
| **Inducing Points** | Med | Med | Med-High |
| **VI** | Med–Low | High | Med-High |
| **PriorCVAE** | Med | High | Med |
| **RFF** | Med-low | Med | Med-High |
| **DeepRV (Ours)** | High | High | Med |

### 3.3 Low-rank Factorizations

A complementary approach to inducing points for scaling GPs is based on low-rank factorizations of the covariance matrix. The core idea is that many kernels have covariance matrices that are approximately low rank, particularly when the input data is smooth or lies on a low-dimensional data manifold. The Nyström method [Williams and Seeger, 2001] exploits a subset of columns of the kernel matrix to construct a low rank approximation. On the other hand, Rahimi and Recht [2007b] use random Fourier features (RFFs) to approximate shift-invariant kernels via Monte Carlo features drawn from the spectral density. These approaches reduce the $\mathcal{O}(N^3)$ to $\mathcal{O}(NM^2)$ or $\mathcal{O}(ND)$ where $M$ is the number of basis functions and $D$ is the number of random features. While highly scalable, these techniques are often less accurate than full GPs or INLA [Heaton et al., 2018].

### 3.4 Variational Inference

Variational inference (VI) provides a scalable alternative to MCMC by approximating a computationally expensive posterior with a parametric distribution drawn from a restricted family. Rather than sampling from the exact posterior, VI fits this distribution by minimizing the KL divergence between the variational distribution and the true posterior. Commonly, the variational distribution is parametrized as a multivariate Gaussian [Hensman et al., 2015, Matthews et al., 2017]. Modern probabilistic programming frameworks enable flexible VI methods, such as Automatic Differentiation Variational Inference (ADVI), which can be applied to GP models without model-specific derivations [Kucukelbir et al., 2017, Phan et al., 2019]. While VI scales well and integrates naturally with probabilistic models, its accuracy is inherently limited by the expressiveness of the variational family.

### 3.5 Composite Likelihood Approximation

Composite likelihood methods scale GPs by replacing the full joint likelihood with a product of low-dimensional conditional distributions. A prominent approach is the Vecchia approximation [Vecchia, 1988], which factorizes to:

$$p(\mathbf{f}) \approx \prod_{i=1}^{N} p\big(f_i \mid \mathbf{f}_{C(i)}\big),$$

where each conditioning set $C(i)$ contains only a small number of previously ordered locations. This induces a sparse, full-rank directed graphical structure and reduces computational complexity to $\mathcal{O}(Nm^2)$, with $m \ll N$. Subsequent work has shown that careful ordering and grouping strategies can substantially improve accuracy [Guinness, 2018, Katzfuss and Guinness, 2021]. However, despite being able to achieve great accuracy, these methods are inherently sequential which hinders scalability.

### 3.6 Neural Surrogates

Recent literature proposes neural surrogates for GPs that can be used as a drop-in replacement in inference frameworks, and include PriorVAE, PriorCVAE, and $\pi$VAE [Semenova et al., 2022, 2023, Mishra et al., 2022]. All of these techniques share a common foundation in Variational autoencoders (VAEs). The fundamental idea of the VAE is that a collection of unknown latent variables control the target data generating process. When the prior on these latents is Gaussian, this is also known as a deep latent Gaussian model (DLGM) [Murphy, 2023].

The objective of these neural surrogates is to train a VAE that can generate samples from a Gaussian process prior. PriorVAE and PriorCVAE, the conditional variant, use a standard MLP-based encoder and decoder. Once the model has been trained, the decoder can generate samples from the prior by decoding a random latent vector, $\mathbf{z}$, and optional conditioning variables, such as the lengthscale and variance.

These techniques, while fast and flexible, suffer from poor accuracy, largely due to the weaknesses associated with VAE-based architectures, such as posterior collapse and oversmoothing. Posterior collapse occurs during encoding, when the model ignores parts of the latent space and produces uninformative representations. Oversmoothing occurs during decoding, when the generated samples are excessively smooth. Furthermore, these errors compound: an error in approximating the latent distribution is exacerbated by a lossy decoding process. These limitations motivated the design of DeepRV, which we detail next.

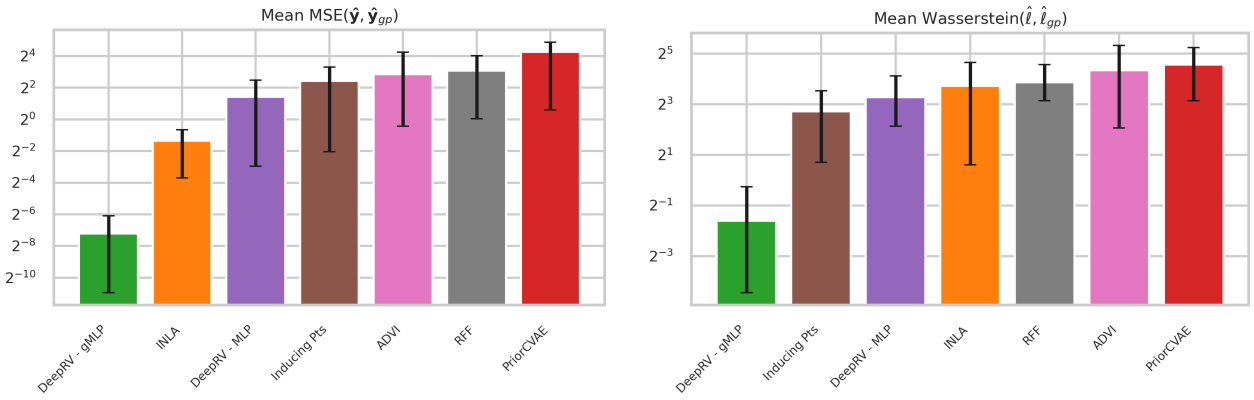


Figure 3: Matérn-1/2 benchmarking results: (a) MSE of each model's posterior mean $\hat{\boldsymbol{y}}$ relative to the full GP MCMC posterior mean $\hat{\boldsymbol{y}}_{\text{gp}}$; (b) Wasserstein distance between each model's posterior over lengthscale $\hat{\ell}$ and full GP MCMC posterior over lengthscale $\hat{\ell}_{\text{gp}}$. Y axis is $\log_2$-scaled, to allow a clear comparison of all methods. Results are averaged across true lengthscales and grid sizes over 15 runs, with 10% and 90% quantiles reported.

# 4 DEEPRV

## 4.1 Method

We introduce **DeepRV**, a highly accurate neural surrogate for Gaussian process realizations. DeepRV differs from previously described neural surrogates in three principal ways: (1) it eliminates the encoding process entirely, (2) it has no information bottleneck, and (3) it leverages factorized stochastic processes directly for training.

These changes hinge on a key insight: for any stochastic process that decomposes into a latent random vector and a linear transformation, the encoding step can be entirely avoided, removing a source of error and allowing the model to focus on accurate decoding. This structure holds naturally for Gaussian processes, since realizations $f(x) \sim \mathcal{GP}(\mu(x), k_\theta(x, x'))$ as in Equations (1)-(3) can be sampled as follows:

$$\begin{aligned} \mathbf{z} &\sim \mathcal{N}(0, \mathbf{I}) \\ L &= \textbf{Cholesky}(K) \\ \mathbf{f} &:= \boldsymbol{\mu} + L\mathbf{z} \end{aligned} \tag{4}$$

Thus, with a known $\mathbf{z}$ and $\mathbf{f}$, we can train a network to learn $L$ and sample from the GP directly. This process is depicted visually in Figure 2.

## 4.2 Architectures

In the following, we present 3 architecture variants for DeepRV: (1) a vanilla MLP, (2) a gated MLP (gMLP) [Liu et al., 2021], and (3) a transformer [Vaswani et al., 2017].

### 4.2.1 MLP

A multilayer perceptron (MLP) consists of sequential layers performing linear transformations followed by nonlinear activations. Stacking multiple layers enables the network to approximate complex, nonlinear mappings. For DeepRV, we use a simple two-layer MLP without dimensionality reduction and with ReLU activations. This maintains consistency with PriorCVAE and highlights that the performance gain achieved from the novel training procedure and decoder-only design, and not only the architectural complexity.

### 4.2.2 gMLP

Gated multilayer perceptrons extend standard MLPs by introducing a gating mechanism [Liu et al., 2021]. If $X \in \mathbb{R}^{N \times D}$ where $N$ is the number of observations or locations and $D$ is an embedding dimension, each gMLP block can be represented by the following equations:

$$Z = \sigma(XU), \quad \tilde{Z} = \text{spatial-gate}(Z), \quad Y = \tilde{Z}V \tag{5}$$

where $U$ and $V$ are trainable linear projections and $\sigma$ is a nonlinearity such as a GELU. The gating function splits $Z$ into two along the channel dimension, yielding $Z_1$ and $Z_2$. $Z_2$ is then projected with learnable $W$ and $b$ and gated by $Z_1$, i.e.

$$Z_1, Z_2 = \text{split}(Z), \quad \tilde{Z} = Z_1 \odot (WZ_2 + b) \tag{6}$$

Gated MLPs are similar to transformer blocks in that they intersperse an attention-like mechanism, i.e. spatial gating, with a feedfoward network. The benefit of this architecture is that it can leverage highly optimized general matrix multiplication (GEMM) operations on the GPU, making it extremely fast to train.

| Metric | N | GP | INLA | Inducing Pts | RFF | PriorCVAE | DeepRV-MLP | DeepRV-gMLP |
|---|---|---|---|---|---|---|---|---|
| MSE($\hat{\mathbf{y}}_{gp}, \hat{\mathbf{y}}$) | 256 | - | 0.392 ± 0.20 | 4.802 ± 2.87 | 7.363 ± 2.98 | 8.064 ± 4.37 | 1.116 ± 0.53 | **0.002 ± 0.00** |
| | 576 | - | 0.333 ± 0.04 | 5.011 ± 1.93 | 9.790 ± 3.87 | 13.470 ± 8.89 | 2.373 ± 0.66 | **0.005 ± 0.00** |
| | 1024 | - | 0.411 ± 0.01 | 6.436 ± 1.65 | 9.735 ± 2.40 | 17.752 ± 4.42 | 4.895 ± 1.46 | **0.009 ± 0.00** |
| | 2,304 | - | 0.261 ± 0.11 | 8.678 ± 5.97 | 11.296 ± 5.33 | 9.877 ± 4.39 | 3.714 ± 1.82 | **0.013 ± 0.01** |
| | 4,096 | - | 0.570 ± 0.49 | 2.158 ± 1.83 | 4.083 ± 3.38 | 47.949 ± 46.96 | 1.367 ± 1.13 | **0.005 ± 0.00** |
| Wass($\hat{\ell}_{gp}, \hat{\ell}$) | 256 | - | 11.68 ± 4.29 | 4.87 ± 1.57 | 11.72 ± 4.49 | 15.66 ± 9.79 | 9.23 ± 2.74 | **0.13 ± 0.08** |
| | 576 | - | 11.74 ± 5.08 | 5.40 ± 2.31 | 13.81 ± 4.57 | 25.62 ± 7.55 | 9.36 ± 1.53 | **0.21 ± 0.06** |
| | 1024 | - | 13.59 ± 7.94 | 9.95 ± 5.16 | 15.72 ± 5.83 | 26.64 ± 10.05 | 12.91 ± 6.29 | **0.26 ± 0.08** |
| | 2,304 | - | 12.90 ± 5.92 | 7.16 ± 2.87 | 15.09 ± 3.65 | 27.33 ± 9.15 | 12.20 ± 3.71 | **0.44 ± 0.36** |
| | 4,096 | - | 16.19 ± 7.37 | 5.61 ± 2.18 | 16.33 ± 4.08 | 23.40 ± 7.03 | 4.87 ± 1.60 | **0.61 ± 0.48** |
| ESS($\ell$)/sec | 256 | 14.38 ± 4.23 | - | **27.76 ± 8.15** | 0.00 ± 0.00 | **56.14 ± 30.51** | **37.32 ± 10.11** | 21.30 ± 6.13 |
| | 576 | 3.19 ± 0.61 | - | 11.70 ± 5.65 | 0.00 ± 0.00 | **13.97 ± 3.31** | **14.34 ± 1.71** | 8.14 ± 1.15 |
| | 1024 | 1.33 ± 0.49 | - | **8.10 ± 4.68** | 0.00 ± 0.00 | **11.98 ± 5.13** | 0.87 ± 0.36 | 6.47 ± 2.44 |
| | 2,304 | 0.35 ± 0.06 | - | 3.97 ± 1.54 | 0.02 ± 0.01 | **8.20 ± 2.85** | 2.32 ± 0.32 | 3.26 ± 0.67 |
| | 4,096 | 0.13 ± 0.03 | - | **2.82 ± 0.94** | 0.01 ± 0.01 | **2.99 ± 1.46** | 0.84 ± 0.12 | **2.74 ± 0.60** |
| Infer Time (s) | 256 | 274 ± 79.89 | **2 ± 0.08** | 154 ± 31.56 | 1,314 ± 127.31 | 81 ± 15.86 | 98 ± 21.46 | 157 ± 51.35 |
| | 576 | 949 ± 257.32 | **4 ± 0.06** | 316 ± 64.61 | 373 ± 36.30 | 171 ± 8.58 | 188 ± 8.05 | 332 ± 95.52 |
| | 1024 | 2,546 ± 805.77 | **7 ± 0.07** | 566 ± 184.98 | 1,028 ± 175.29 | 231 ± 13.62 | 309 ± 52.97 | 510 ± 177.81 |
| | 2,304 | 7,476 ± 1,428.08 | **38 ± 1.53** | 862 ± 195.73 | 1,653 ± 501.73 | 334 ± 60.64 | 394 ± 23.93 | 778 ± 242.11 |
| | 4,096 | 20,659 ± 3,887.21 | **95 ± 1.88** | 955 ± 177.34 | 3,848 ± 1,242.73 | 595 ± 111.21 | 974 ± 170.98 | 939 ± 169.77 |

Table 2: Matérn-1/2 benchmarking results: (a) Posterior predictive MSE relative to full GP MCMC; (b) Wasserstein distance between inferred and full GP MCMC lengthscale posteriors; (c) Effective $\ell$ sample size (ESS) per second; (d) Inference time in seconds. Results are shown for each grid size and are averaged across the three true lengthscales (10, 30, 50) over 15 runs, with the standard error reported.

A downside, however, is that the number of tokens or locations is fixed. For DeepRV, we use a simple two-layer gMLP without an information bottleneck.

#### 4.2.3 Transformer

To handle variably sized and arbitrarily chosen spatial locations (Section 6.4), we employ a transformer-based DeepRV decoder. Transformers, originally introduced for sequence modelling [Vaswani et al., 2017], consist of stacked multi-headed attention and feedforward layers with residual connections. We use a similar approach with two main modifications. First, we include ID positional embeddings to encode an ordering over locations. Concretely, the input token at index $i$ is given by

$$\mathbf{x}_i \;=\; \mathbf{z}_i \,\|\, \mathrm{embed}_s(\mathbf{s}_i) \,\|\, \mathrm{embed}_{\mathrm{id}}(i), \qquad (7)$$

where $\mathbf{z}_i$ is the latent input, $\mathbf{s}_i$ denotes the spatial location, $\mathrm{embed}_s(\cdot)$ is a spatial embedding (*e.g.*, identity or RFF positional encodings), and $\mathrm{embed}_{\mathrm{id}}(i)$ is a learned ID embedding associated with the input index. Without ID embeddings, the model struggled to learn with arbitrary input locations. We hypothesize that these embeddings help the transformer implicitly represent the Cholesky factor $L$, whose structure depends on the ordering of inputs. Second, we incorporate a kernel-based attention bias [Jenson et al., 2026]. The biased attention is defined as

$$\mathcal{K}(\mathbf{Q},\mathbf{K})\mathbf{V} \;\coloneqq\; \mathrm{softmax}\!\left(\frac{\mathbf{Q}\mathbf{K}^{\intercal}}{\sqrt{d_k}} \;+\; \alpha\,\mathbf{K}_{\boldsymbol{\theta}}\right)\mathbf{V}, \qquad (8)$$

where $\mathbf{K}_{\boldsymbol{\theta}}$ is the GP kernel matrix conditioned on hyperparameters $\boldsymbol{\theta}$, and $\alpha$ is a learnable scalar per attention head. This bias injects prior GP structure into attention weights and improves the fidelity with which the network reconstructs GP realisations.

## 5 DATA GENERATION, PRE-TRAINING, AND INFERENCE

In order to train DeepRV, a dataset consisting of tuples of $(\boldsymbol{\theta}, \mathbf{z}, \mathbf{f})$ is created according to the following process:

1. Sample kernel parameters: $\boldsymbol{\theta} \sim p_\theta(\cdot)$.
2. Materialize the kernel: $K = k_\theta(\mathbf{x}, \mathbf{x}')$.
3. Decompose the kernel: $L = \mathrm{Cholesky}(K)$.
4. Sample random normal vector: $\mathbf{z} \sim \mathcal{N}(0, I_d)$.
5. Generate the function values: $\mathbf{f} = L\mathbf{z}$.

The input to DeepRV is $(\boldsymbol{\theta}, \mathbf{z})$ and it outputs an estimate of function values $\hat{\mathbf{f}}$. The loss function is MSE between $\hat{\mathbf{f}}$ and the true $\mathbf{f}$.

Once trained, DeepRV can map a latent random vector, $\mathbf{z}$, and kernel parameters, $\boldsymbol{\theta}$, to an instance of the target stochastic process conditioned on those parameters. Accordingly, inside a probabilistic programming language like NumPyro, sampling from a GP can be replaced with sampling a random normal vector, $\mathbf{z}$, and passing $(\boldsymbol{\theta}, \mathbf{z})$ through DeepRV in order to generate the sample $\hat{\mathbf{f}}$. This process is detailed in Figure 2.

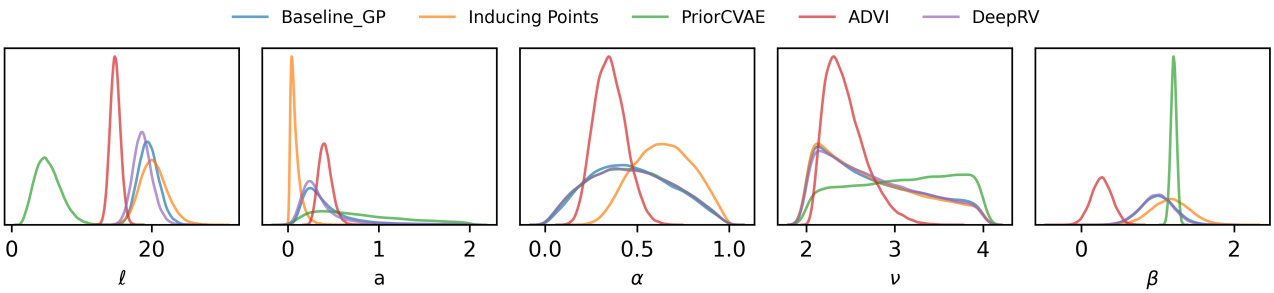


Figure 4: Spatiotemporal GP inferred hyperparameter posterior distributions. DeepRV closely matches GP on all hyperparameter posterior distributions.

## 6 EXPERIMENTS

In this section, we present a set of experiments that evaluate DeepRV's fidelity to full GP inference and its scalability by benchmarking against a range of existing methods. We further demonstrate its flexibility by performing inference with a non-separable spatiotemporal kernel, evaluate its applicability to real-world data, and introduce an extended variant capable of operating on arbitrary input locations.

All code required to reproduce these experiments is available in `dl4bi` Python package.

### 6.1 Benchmarking DeepRV

We simulated data over 2D grids of increasing resolution, $N = 16^2$, $24^2$, $32^2$, $48^2$, and $64^2$, to assess the scalability and accuracy of DeepRV in spatial inference. We benchmarked DeepRV against INLA, Inducing Points, PriorCVAE, RFFs, and ADVI. INLA was tested using the standard `R-INLA` package, with meshes scaled to resolution, using the Laplace approximation with grid-based integration for accuracy. We selected $N^{2/3}$ anchors for the Inducing points method, and $2L$ features for RFF to match DeepRV's complexity for fairness. For ADVI, we used NumPyro's `AutoMultivariateNormal` guide to implement a full-rank Gaussian posterior. We refer the reader to Appendix A for implementation details of the benchmarking methods, and Appendix B.6 for a direct comparison between Vecchia approximations and DeepRV which were perform separately.

Grid coordinates were normalized to $[0, 100]$ and used as GP inputs. DeepRV and PriorCVAE were trained to emulate a Matérn-1/2 GP prior, with mini-batches of 32 for 200K steps (300K for $48^2$ and $64^2$ grids). The lengthscale $\ell$ was drawn from a LogNormal$(3.0, 0.4)$ prior (consistent with R-INLA mesh settings), and the variance fixed at 1, since the data can always be standardized prior to inference. After training, the learned priors were used in a NumPyro inference model with a Poisson likelihood:

$$\begin{aligned} \boldsymbol{\theta} :=& \ell, \sigma \sim p_{\boldsymbol{\theta}}(\cdot), \\ \mathbf{f}_{\boldsymbol{\theta}} \sim& \mathcal{GP}_{\boldsymbol{\theta}}(\cdot), \\ \beta \sim& \mathcal{N}(0, 1000), \\ \boldsymbol{\lambda} =& \exp(\beta + \mathbf{f}_{\boldsymbol{\theta}}), \\ \mathbf{y} \sim& \text{Poisson}(\boldsymbol{\lambda}). \end{aligned} \tag{9}$$

For inference we used NUTS [Hoffman and Gelman, 2014] with two chains for grids $\leq 32^2$ and one chain for larger grids. While running a single chain is unusual, for the GP baseline at the largest grids one chain can take tens of hours, so this trade-off was necessary to make benchmarking feasible. We ran 4,000 warmup steps and 6,000 posterior draws per chain. Observations were generated with true $\beta = 1.5$ and $\ell \in \{10, 30, 50\}$, with approximately 50% masked in contiguous regions to increase difficulty.

Results for the Matérn-1/2 kernel are presented in Table 2 and Figure 3. We also repeated the experiment with a Matérn-3/2 kernel, which is not supported by standard `R-INLA` package for 2D inputs. Results are provided in Appendix Table 7 and Figure 6.

Across settings, DeepRV achieves the highest fidelity to full GP inference in both predictive performance and hyperparameter recovery. INLA is consistently the fastest and provides competitive predictive accuracy, but weaker parameter inference. PriorCVAE yields the highest effective sample size per second, yet this is misleading since its predictive and parameter accuracy are among the lowest, highlighting ESS/sec as an incomplete standalone measure.

### 6.2 DeepRV flexibility: non-separable spatiotemporal kernel

To demonstrate DeepRV's flexibility relative to other GP approximation methods, we performed inference

using a non-separable space–time covariance function inspired by Gneiting [Gneiting, 2002], defined as

$$k_{\boldsymbol{\theta}}(\mathbf{s}, t; \mathbf{s}', t') = \frac{\sigma^2}{(a d_t^{2\alpha} + 1)^{\nu}} \exp\left( - \frac{\|\mathbf{s} - \mathbf{s}'\|^2}{\ell^2 (a d_t^{2\alpha} + 1)^{b}} \right)$$

where $d_t = |t - t'|$, and the hyperparameters are $\boldsymbol{\theta} := \{\ell, \sigma^2, a, \alpha, b, \nu\}$. This kernel captures both spatial and temporal correlations in a non-separable manner.

Such genuinely non-separable structures are typically not directly supported in default INLA workflows and require specialised model formulations [Bakka et al., 2018], and they cannot be handled by standard RFF approximations.

We followed the training and inference procedure described in Section 6.1, with the only changes being the hyperparameter set $\boldsymbol{\theta}$ above, a single spatial grid of size $16^2$ with 5 time steps, and we trained the neural networks for 500,000 steps. We set the hyperparameters to $\sigma^2 = 1.0, \ell = 20.0, \beta = 1.0, a = 0.5, \alpha = 0.8, b = 1.0, \nu = 1.0$.

Spatial masking was applied as before, with ≈50% of observations masked in contiguous regions, consistent across all time steps. Additionally, observations at $t = 2, 3$ were removed to simulate partially observed temporal dynamics. The resulting inferred hyperparameter distributions are shown in Figure 4, and the posterior predictive across time is presented in Figure 7. The results demonstrate that DeepRV matches GP predictive performance and parameter inference even in settings with more hyperparameters and complex interdependencies. This flexibility arises from DeepRV's simple design, which does not rely on structural assumptions about the GP it emulates.

### 6.3 Real-world application: London LSOA

We applied DeepRV to the education dimension of deprivation in London across 4,994 LSOAs. A household is deprived if no member has at least level 2 education and no one aged 16–18 is a full-time student. Data was taken from the ONS dataset generator[1], with boundaries from the ONS Open Geography Portal[2]

For validation, we also fit (i) a full GP at the MSOA level ($n$ = 1,024), where exact inference is still feasible, and (ii) a short full-GP run at the LSOA level (2 chains, 1,000 warmup, 500 posterior samples) to calibrate against DeepRV. This lets us check that DeepRV at both resolutions is consistent with a GP baseline.

We ran 4 chains with 4,000 warmup and 4,000 posterior samples (as in Section 6.1). To assess robustness, we randomly masked 50% of observations. LSOA-level predictive means are shown in Figure 1. Model-vs-model comparisons of predicted prevalence (DeepRV vs. GP) are shown in Figure 5 (MSOA) and Appendix Figure 10 (LSOA). Comparisons against observed prevalence at unobserved MSOA locations are provided in Figure 8 and Figure 9. We modelled the number of deprived households using a simple binomial likelihood:

$$\begin{aligned}
\boldsymbol{\theta} :=& \ell, \sigma \sim p_{\boldsymbol{\theta}}(\cdot), \\
& \mathbf{z} \sim \mathcal{N}(0, \mathbf{I}), && (10) \\
\mathbf{f}_{\boldsymbol{\theta}} =& \text{DeepRV}(\mathbf{z}, \boldsymbol{\theta}), && (11) \\
& \beta \sim \mathcal{N}(0, 1), \\
\mathbf{p} =& \text{logit}^{-1}(\beta + \mathbf{f}_{\boldsymbol{\theta}}), && (12) \\
\mathbf{y} \sim& \text{Binomial}(\mathbf{N}, \mathbf{p}),
\end{aligned}$$

where $\mathbf{N}$ denotes the number of households in each LSOA. Across these checks, DeepRV closely matches the GP in both predicted prevalence and uncertainty on this real-world dataset. A full LSOA GP run would require approximately ∼70 hours on our hardware, whereas DeepRV completed in about 3 hours, enabling high-fidelity inference at city scale.

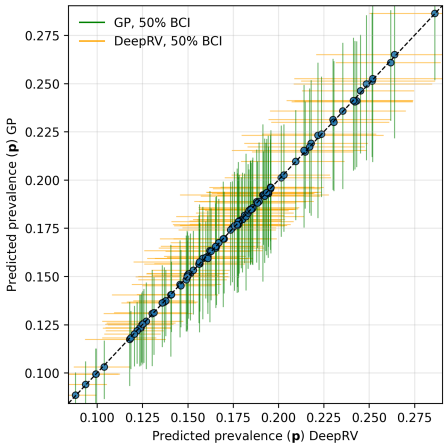


Figure 5: Predicted prevalence at 100 randomly selected MSOAs.

### 6.4 Arbitrary Locations DeepRV

We next assess DeepRV's ability to generalize across datasets with varying numbers and arbitrarily chosen locations. In this setting, both the placement of locations is arbitrary (uniformly sampled) and the number of inputs can change. To emulate GPs in this setting, we use the transformer-based DeepRV, which naturally handles variable-length inputs. As mentioned in Section 4.2.3, we employ ID positional embeddings,

[1] https://www.ons.gov.uk/filters/dcf91941-1de3-4e26-8cae-4adec2a42f9c/dimensions

[2] https://geoportal.statistics.gov.uk/

which in turn requires to specify a priori the maximum number of locations that can be inferred. This design makes it possible to train once and then apply the model to any new set of locations, up to the specified maximum.

We follow the same Matérn-1/2 kernel and Poisson likelihood setup as in subsection 6.1, but increase model capacity to four layers, add RFF positional encodings and train for 2M steps. Inference was then performed on three datasets of randomly sampled locations in $[0, 100]$ with $N = 512$, 1024, and 2048, benchmarking against both a GP and inducing points.

The results in Table 3 show that DeepRV closely matches the GP baseline across predictive and parameter metrics on arbitrary locations. However, to handle this more complex task, the transformer is larger and slower, yielding only modest speed gains (≈10%). Additional information, including posterior distribution comparisons, and positional encodings ablations that underscores the importance of the ID embeddings for this experiment are provided in Appendix subsection B.4.

| Model | MSE($\hat{\mathbf{y}}_{gp}, \hat{\mathbf{y}}$) | Wass($\hat{\ell}_{gp}, \hat{\ell}$) | LPD | Cover-80% |
|---|---|---|---|---|
| GP | - | - | **-2.00 ± 0.08** | **0.97 ± 0.01** |
| DeepRV | **0.01 ± 0.01** | **0.66 ± 0.20** | **-2.00 ± 0.08** | **0.97 ± 0.01** |
| Inducing Pts | 1.82 ± 1.10 | 3.88 ± 0.15 | -2.09 ± 0.10 | 0.86 ± 0.02 |

Table 3: Arbitrary-locations experiment results: (a) Posterior predictive MSE relative to GP; (b) Wasserstein distance between inferred and GP lengthscale posteriors; (c) Log predictive density (LPD); (d) Coverage of the 80% posterior predictive. Results are averaged across dataset sizes, with the standard error reported.

Table 4: Architecture ablation for accuracy metrics.

| Model | MSE($\hat{\mathbf{y}}_{gp}, \hat{\mathbf{y}}$) | Test Loss | Wass($\hat{\ell}_{gp}, \hat{\ell}$) |
|---|---|---|---|
| PriorCVAE | 4.424 ± 1.332 | 0.1610 ± 0.0187 | 18.342 ± 4.673 |
| DeepRV-MLP | 0.399 ± 0.143 | 0.0430 ± 0.0017 | 5.545 ± 1.014 |
| DeepRV-gMLP | **0.005 ± 0.002** | **0.001 ± 0.000** | **0.308 ± 0.088** |
| DeepRV-Trans | 0.033 ± 0.012 | 0.004 ± 0.000 | 0.921 ± 0.320 |

Table 5: Architecture ablation for efficiency metrics.

| Model | ESS($\ell$)/sec | Infer Time (s) | Train Time (s) |
|---|---|---|---|
| GP | 5.035 ± 0.553 | 265.94 ± 26.64 | - |
| PriorCVAE | **36.174 ± 5.475** | **60.94 ± 3.19** | 150.6 ± 0.29 |
| DeepRV-MLP | 7.536 ± 1.040 | 73.03 ± 7.25 | **128.5 ± 0.32** |
| DeepRV-gMLP | 12.709 ± 1.324 | 102.66 ± 9.29 | 283.7 ± 0.42 |
| DeepRV-Trans | 6.474 ± 0.684 | 156.83 ± 12.91 | 2200.5 ± 0.50 |

### 6.5 Ablation Study

We evaluate our architectural choices in DeepRV by comparing DeepRV–MLP, DeepRV–gMLP, DeepRV–Transformer with kernel attention, PriorCVAE, and a GP. We followed the same setup as subsection 6.1 on a fixed 512-point 2D grid, across four kernels (Matérn 1/2, 3/2, 5/2, RBF) and three random seeds.

The results in Table 4 and Table 5 show that the performance gap between PriorCVAE and DeepRV–MLP stems from the decoder-only design rather than architectural complexity, as both use the same MLP backbone. The transformer variant matches gMLP accuracy but at much higher computational cost, restricting it to the variable-location setting. Overall, all DeepRV variants approximate full GP inference well, with gMLP offering the best trade-off between accuracy and efficiency.

## 7 CONCLUSION

We presented **DeepRV**, a decoder-only neural surrogate for Gaussian processes that maps latent draws and kernel parameters directly to function values. Across simulated spatial benchmarks, a non-separable spatiotemporal setting, and a city-scale application (London LSOA), DeepRV consistently matched full GP inference in both predictions and hyperparameter recovery while substantially accelerating MCMC-based inference and retaining modelling flexibility. Compared with popular scalable alternatives (INLA, inducing points, RFFs, VAEs), DeepRV offered the strongest fidelity to exact GPs at practical runtimes on a single GPU, and extended naturally to a transformer variant. The main limitations are the cubic pre-training cost to generate supervision and the assumption of a deterministic mapping from randomness to outputs. Future work will focus on reducing pre-training cost (*e.g.*, Flash or Flex Attention [Dao et al., 2022, Dao, 2024, Shah et al., 2024, Dong et al., 2024]), improving transfer to unseen resolutions and geometries, and extending the paradigm to broader classes of stochastic processes and probabilistic simulators.

### Acknowledgments

J.N. and E.S. acknowledge support in part by the AI2050 program at Schmidt Sciences (Grant [G-22-64476]). J.N. and S.F. acknowledge the EPSRC (EP/V002910/2). D.J. acknowledges his Google DeepMind scholarship. We thank Tom Rainforth for advice on the method and James Bennett for advice on data access. We thank Paolo Andrich and Samir Bhatt for their feedback on the manuscript.

### References

Vasiliki D. Agou, A. Pavlides, and Dionissios T. Hristopulos. Spatial modeling of precipitation based

on data-driven warping of gaussian processes. *Entropy*, 24(3):321, 2022. doi: 10.3390/e24030321.

Fabian E. Bachl, Finn Lindgren, David L. Borchers, and Janine B. Illian. inlabru: an r package for bayesian spatial modelling from ecological survey data. *Methods in Ecology and Evolution*, 10(6):760–766, 2019. doi: https://doi.org/10.1111/2041-210X.13168. URL https://besjournals.onlinelibrary.wiley.com/doi/abs/10.1111/2041-210X.13168.

Haakon Bakka, Håvard Rue, Geir-Arne Fuglstad, Andrea Riebler, David Bolin, Elias Krainski, Daniel Simpson, and Finn Lindgren. Spatial modelling with r-inla: A review. *WIREs Computational Statistics*, 10(6):e1443, 2018.

Y. Cheng, Xiucheng Li, Zhijun Li, Shouxu Jiang, and Xiaofan Jiang. Fine-grained air quality monitoring based on gaussian process regression. In *Neural Information Processing: 21st International Conference*, page 126–134, 2014.

Lehel Csató and Manfred Opper. Sparse online gaussian processes. *Neural Computation*, 14(3):641–668, 2002. doi: 10.1162/089976602317250933.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Aadesh Desai, Eshan Gujarathi, Saagar Parikh, Sachin Yadav, Zeel Patel, and Nipun Batra. Deep gaussian processes for air quality inference. *arXiv preprint*, 2022. arXiv:2211.10174.

J. Diggle, P. and Emanuele Giorgi. Model-based geostatistics for prevalence mapping in low-resource settings. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 2015. URL https://arxiv.org/abs/1505.06891. preprint arXiv:1505.06891.

J. Diggle, P. A. Tawn, J. and A. Moyeed, R.˙ Model-based geostatistics. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 47(3):299–350, 1998. doi: 10.1111/1467-9876.00113.

J. Diggle, P. Paula Moraga, Barry R. Rowlingson, and Benjamin M. Taylor. Spatial and spatio-temporal log–gaussian cox processes: Extending the geostatistical paradigm. *arXiv preprint*, 2013. URL https://arxiv.org/abs/1312.6536. arXiv:1312.6536.

Juechu Dong, Boyuan Feng, Driss Guessous, Yanbo Liang, and Horace He. Flex attention: A programming model for generating optimized attention kernels, 2024. URL https://arxiv.org/abs/2412.05496.

Tilmann Gneiting. Nonseparable, stationary covariance functions for space–time data. *Journal of the American Statistical Association*, 97(458):590–600, 2002. doi: 10.1198/016214502760047113. URL https://doi.org/10.1198/016214502760047113.

Virgilio Gómez-Rubio and Ferrán Palmí-Perales. Spatial models with the integrated nested laplace approximation within markov chain monte carlo. *arXiv preprint arXiv:1702.03891*, 2017. URL https://arxiv.org/abs/1702.03891.

Joseph Guinness. Permutation and grouping methods for sharpening gaussian process approximations. *Technometrics*, 60(4):415–429, June 2018. ISSN 1537-2723. doi: 10.1080/00401706.2018.1437476. URL http://dx.doi.org/10.1080/00401706.2018.1437476.

Matthew J. Heaton, Abhirup Datta, Andrew Finley, Reinhard Furrer, Rajarshi Guhaniyogi, Florian Gerber, Robert B. Gramacy, Dorit Hammerling, Matthias Katzfuss, Finn Lindgren, Douglas W. Nychka, Furong Sun, and Andrew Zammit-Mangion. A case study competition among methods for analyzing large spatial data, 2018. URL https://arxiv.org/abs/1710.05013.

James Hensman, Nicoló Fusi, and Neil D. Lawrence. Gaussian processes for big data. In *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 282–290, 2013.

James Hensman, Alexander G. de G. Matthews, and Zoubin Ghahramani. Scalable variational gaussian process classification. *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 38:351–360, 2015.

Matthew D Hoffman and Andrew Gelman. The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, 15(1):1593–1623, 2014.

Daniel Jenson, Jhonathan Navott, Piotr Grynfelder, Mengyan Zhang, Makkunda Sharma, Elizaveta Semenova, and Seth Flaxman. Scalable spatiotemporal inference with biased scan attention transformer neural processes. In *Proceedings of the 29th International Conference on Artificial Intelligence and Statistics*, volume 300 of *Proceedings of Machine Learning Research*, Tangier, Morocco, 2026. PMLR.

Matthias Katzfuss and Joseph Guinness. A general framework for vecchia approximations of gaussian processes. *Statistical Science*, 36(1), February 2021.

ISSN 0883-4237. doi: 10.1214/19-sts755. URL http://dx.doi.org/10.1214/19-STS755.

Marlene Klockmann, Udo von Toussaint, and Eduardo Zorita. Towards variance-conserving reconstructions of climate indices with gaussian process regression in an embedding space. *Geoscientific Model Development*, 17(5):1765–1787, 2024. doi: 10.5194/gmd-17-1765-2024.

Jonathan Koh, François Pimont, Jean-Luc Dupuy, and Thomas Opitz. Spatiotemporal wildfire modeling through point processes with moderate and extreme marks. *arXiv preprint*, 2021. arXiv:2105.08004.

Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David M. Blei. Automatic differentiation variational inference. *Journal of Machine Learning Research*, 18(14):1–45, 2017. URL http://jmlr.org/papers/v18/16-107.html.

Andrew B. Lawson. *Bayesian Disease Mapping: Hierarchical Modeling in Spatial Epidemiology*. Chapman & Hall/CRC Interdisciplinary Statistics. CRC Press, 3rd edition, 2018. ISBN 978-1138030362.

Finn Lindgren, Håvard Rue, and Johan Lindström. An explicit link between gaussian fields and gaussian markov random fields: The stochastic partial differential equation approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(4):423–498, 2011. doi: 10.1111/j.1467-9868.2011.00777.x.

Hanxiao Liu, Zihang Dai, David R. So, and Quoc V. Le. Pay attention to mlps. In *Advances in Neural Information Processing Systems*, volume 34, pages 1–19, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/4cc05b35c2f937c5bd9e7d41d3686fff-Abstract.html.

Karam Mansour, Stefano Decesari, Marco Paglione, Silvia Becagli, others, and Matteo Rinaldi. Nested cross-validation gaussian process to model dimethylsulfide mesoscale variations in warm oligotrophic mediterranean seawater. *npj Climate and Atmospheric Science*, 7(277), 2024. doi: 10.1038/s41612-024-00830-y.

Alexander G. de G. Matthews, James Hensman, Richard E. Turner, and Zoubin Ghahramani. On sparse variational methods and the Kullback–Leibler divergence between stochastic processes. *Journal of Machine Learning Research*, 18(52):1–63, 2017.

Swapnil Mishra, Seth Flaxman, Tresnia Berah, Mikko Pakkanen, Harrison Zhu, and Samir Bhatt. *pi* vae: Encoding stochastic process priors with variational autoencoders. *Statistics & Computing*, 2022.

Kevin P. Murphy. *Probabilistic Machine Learning: Advanced Topics*. MIT Press, 2023. URL http://probml.github.io/book2.

Zeel B. Patel, Palak Purohit, Harsh M. Patel, Shivam Sahni, and Nipun Batra. Accurate and scalable gaussian processes for fine-grained air quality inference. In *Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI-22)*, pages 12080–12088, 2022.

Du Phan, Neeraj Pradhan, and Martin Jankowiak. Composable effects for flexible and accelerated probabilistic programming in numpyro. *arXiv preprint arXiv:1912.11554*, 2019.

Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, pages 548–555, 2005. doi: 10.1145/1102351.1102425.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 20, pages 1177–1184, 2007a.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007b. URL https://proceedings.neurips.cc/paper_files/paper/2007/file/013a006f03dbc5392effeb8f18fda755-Paper.pdf.

Håvard Rue, Sara Martino, and Nicolas Chopin. Approximate bayesian inference for latent gaussian models by using integrated nested laplace approximations. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(2):319–392, 2009. doi: 10.1111/j.1467-9868.2008.00700.x.

Håvard Rue, Andrea Riebler, Sigrunn H. Sørbye, Janine B. Illian, Daniel P. Simpson, and Finn K. Lindgren. Bayesian computing with INLA: A review. *Annual Review of Statistics and Its Application*, 4:395–421, 2017. doi: 10.1146/annurev-statistics-060116-054045.

Håvard Rue et al. *The R-INLA Project: INLA Manual*, 2023. URL https://www.inla.r-inla-download.org/r-inla.org/doc/inla-manual/inla-manual.pdf.

Elizaveta Semenova, Yidan Xu, Adam Howes, Theo Rashid, Samir Bhatt, Swapnil Mishra, and Seth Flaxman. Priorvae: encoding spatial priors with variational autoencoders for small-area estimation. *Journal of the Royal Society Interface*, 19(191):20220094, 2022.

Elizaveta Semenova, Prakhar Verma, Max Cairney-Leeming, Arno Solin, Samir Bhatt, and Seth Flaxman. Priorcvae: scalable mcmc parameter inference with bayesian deep generative modelling. *arXiv preprint arXiv:2304.04307*, 2023.

Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. Flashattention-3: Fast and accurate attention with asynchrony and low-precision. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=tVConYid20.

Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 18, pages 1257–1264, 2006.

Aaron Sonabend, Jiangshan Zhang, Joel Schwartz, Brent A. Coull, and Junwei Lu. Scalable gaussian process regression via median posterior inference for estimating multi-pollutant mixture health effects. *arXiv preprint*, 2024. arXiv:2411.10858.

Clara Stoddart, Lauren Shrack, Richard Sserunjogi, Usman Abdul-Ganiy, Engineer Bainomugisha, Deo Okure, Ruth Misener, Jose Pablo Folch, and Ruby Sedgwick. Gaussian processes for monitoring air-quality in kampala. *arXiv preprint*, 2023. arXiv:2311.16625.

Michalis K. Titsias. Variational learning of inducing variables in sparse gaussian processes. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 567–574, 2009.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

A. V. Vecchia. Estimation and model identification for continuous spatial processes. *Journal of the Royal Statistical Society. Series B (Methodological)*, 50(2): 297–312, 1988. ISSN 00359246. URL http://www.jstor.org/stable/2345768.

Peng Wang, Lyudmila Mihaylova, Rohit Chakraborty, Said Munir, Martin Mayfield, Khan Alam, Muhammad Fahim Khokhar, Zhengkai Zheng, Chengxi Jiang, and Hui Fang. A gaussian process method with uncertainty quantification for air quality forecasting. *Atmosphere*, 12(10):1344, 2021. doi: 10.3390/atmos12101344.

Wen Wang, Quan J. Wang, and Rory Nathan. Gaussian process regression on multiple drivers and attributes for rapid prediction of maximum flood inundation extent and depth. *Journal of Hydrology*, 649: 132476, 2024. doi: 10.1016/j.jhydrol.2024.132476.

Christopher K. I. Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 13, pages 682–688, 2001.

Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

Xiaoyu Xiong, Benjamin D. Youngman, and Theodoros Economou. Data fusion with gaussian processes for estimation of environmental footprints. *Environmetrics*, 32(6):e2660, 2021. doi: 10.1002/env.2660.

Tianjian Zhou and Yuan Ji. Semiparametric bayesian inference for the transmission dynamics of covid-19 with a state-space model. *arXiv preprint*, 2020. URL https://arxiv.org/abs/2006.05581. arXiv:2006.05581.

## Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes.]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes.]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Not Applicable. We provide a publicly available code repository.]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Not Applicable.]
   (b) Complete proofs of all theoretical results. [Not Applicable.]
   (c) Clear explanations of any assumptions. [Not Applicable.]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes. The paper can be fully reproduced through the provided codebase. Link to the repository can be found in the abstract and in Section 6.]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes. Full training details can be found in the appendix section of each experiment, and in the provided codebase.]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes.]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes. This information can be found in the appendix section of each experiment.]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes.]
   (b) The license information of the assets, if applicable. [Yes. We use only open-source codebases and publicly available data. The licenses of third-party software are specified in their respective repositories, our code is released publicly on GitHub under its repository license, and the ONS data are distributed under their stated public terms.]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes. The codebase URL is provided in the paper, and the maps used to generate the real-life data experiment subsection 6.3 can be downloaded using the instructions within the codebase.]
   (d) Information about consent from data providers/curators. [Not Applicable.]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable.]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable.]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable.]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable.]

# A Benchmark GP Approximations

This appendix provides the explicit mathematical formulations of the GP approximations used to benchmark DeepRV.

## A.1 Integrated Nested Laplace Approximation (INLA)

Integrated Nested Laplace Approximation (INLA) [Rue et al., 2009] provides a deterministic alternative to MCMC for latent Gaussian models (LGMs) by approximating posterior marginals.

**Latent Gaussian model.** An LGM with latent field $\mathbf{f}$ and hyperparameters $\boldsymbol{\theta}$ has joint density of:

$$p(\mathbf{y}, \mathbf{f}, \boldsymbol{\theta}) = p(\boldsymbol{\theta})\, p(\mathbf{f} \mid \boldsymbol{\theta}) \prod_{i=1}^{N} p(y_i \mid f_i, \boldsymbol{\theta}), \qquad \mathbf{f} \mid \boldsymbol{\theta} \sim \mathcal{N}\big(\mathbf{0}, \mathbf{Q}(\boldsymbol{\theta})^{-1}\big), \tag{13}$$

where $\mathbf{Q}(\boldsymbol{\theta})$ is sparse.

**INLA approximation.** Let $\tilde{p}_G(\mathbf{f} \mid \boldsymbol{\theta}, \mathbf{y})$ be a Gaussian approximation to $p(\mathbf{f} \mid \boldsymbol{\theta}, \mathbf{y})$ centred at its mode $\mathbf{f}^*(\boldsymbol{\theta})$. INLA approximates the hyperparameter posterior [Rue et al., 2009] by:

$$\tilde{p}(\boldsymbol{\theta} \mid \mathbf{y}) \propto \left. \frac{p(\mathbf{y}, \mathbf{f}, \boldsymbol{\theta})}{\tilde{p}_G(\mathbf{f} \mid \boldsymbol{\theta}, \mathbf{y})} \right|_{\mathbf{f}=\mathbf{f}^*(\boldsymbol{\theta})}, \tag{14}$$

and then computes latent marginals by numerical integration over the low-dimensional $\boldsymbol{\theta}$ [Rue et al., 2009].

**SPDE–GMRF construction.** The stochastic partial differential equation (SPDE) approach represents a Matérn Gaussian field as the stationary solution $f(\mathbf{s})$ to a linear stochastic PDE, enabling a sparse Gaussian Markov random field (GMRF) discretisation on a mesh [Lindgren et al., 2011]:

$$(\kappa^2 - \Delta)^{\alpha/2}\big(\tau f(\mathbf{s})\big) = \mathcal{W}(\mathbf{s}), \qquad \nu = \alpha - \frac{d}{2}. \tag{15}$$

Discretising on a triangular mesh gives

$$f(\mathbf{s}) \approx \sum_{g=1}^{G} \psi_g(\mathbf{s}) w_g, \tag{16}$$

where $\{\psi_g\}$ are local basis functions and $\mathbf{w} = (w_1, \ldots, w_G)$ is a Gaussian vector with sparse precision $\mathbf{Q}(\boldsymbol{\theta})$ induced by local mesh neighbourhoods. For our $d = 2$ grids we set $\alpha = 1.5$, hence $\nu = 0.5$, meaning $f(\mathbf{s})$ is approximating the Matérn-1/2 kernel.

**Code implementation.** We fit a Poisson log-link model

$$y_i \mid \eta_i \sim \text{Poisson}(\lambda_i), \quad \lambda_i = \exp(\eta_i), \quad \eta_i = \beta_0 + f(\mathbf{s}_i), \tag{17}$$

with $\beta_0 \sim \mathcal{N}(0, 1000)$. We parameterise $\boldsymbol{\theta} = (\theta_1, \theta_2) = (\log \tau, \log \kappa)$ and report the inverse-scale lengthscale

$$\ell = \kappa^{-1} = \exp(-\theta_2). \tag{18}$$

We set $\theta_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$ with $(\mu_2, \sigma_2) = (-3.0, 0.8)$ and impose an approximate unit-variance constraint by concentrating

$$\theta_1 + \theta_2 \sim \mathcal{N}\Big( -\tfrac{1}{2}\log(4\pi),\, \epsilon^2 \Big), \qquad \epsilon = 0.05, \tag{19}$$

implemented as a bivariate Gaussian prior on $(\theta_1, \theta_2)$ in `R-INLA`. This prior is constructed so that, under the SPDE parameterisation used in our runs, the implied Matérn marginal variance is approximately fixed to $\sigma^2 \approx 1$ (up to the tolerance $\epsilon$), matching the $\sigma^2 = 1$ convention used in our simulated data. We use `strategy="laplace"` to form $\tilde{p}_G(\mathbf{f} \mid \boldsymbol{\theta}, \mathbf{y})$ via a Laplace approximation around $\mathbf{f}^*(\boldsymbol{\theta})$, and `int.strategy="grid"` to integrate over $\boldsymbol{\theta}$ on a deterministic grid when computing marginals.

### A.2 Inducing Point Approximation

We benchmark DeepRV against a classical subset-of-regressors (SoR), also known as the deterministic training conditional (DTC), inducing point approximation [Quiñonero-Candela and Rasmussen, 2005, Snelson and Ghahramani, 2006].

Let $\mathbf{x} \in \mathbb{R}^N$ denote observation locations and $\mathbf{u} \in \mathbb{R}^M$ inducing locations with $M \ll N$. Define inducing variables

$$\mathbf{f}_u \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_{uu}),$$

where $\mathbf{K}_{uu} = k_\theta(\mathbf{u}, \mathbf{u})$. The latent GP values at $\mathbf{x}$ are approximated as

$$\mathbf{f} \approx \mathbf{K}_{xu}\mathbf{K}_{uu}^{-1}\mathbf{f}_u,$$

with $\mathbf{K}_{xu} = k_\theta(\mathbf{x}, \mathbf{u})$.

**Code implementation.** Within our MCMC model, we sample

$$\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}),$$

and compute

$$\mathbf{L}_{uu} = \text{Cholesky}(\mathbf{K}_{uu}), \quad \bar{\mathbf{f}}_u = \mathbf{L}_{uu}^{-\top}\mathbf{z}, \quad \mathbf{f} = \mathbf{K}_{xu}\bar{\mathbf{f}}_u.$$

This is algebraically equivalent to $\mathbf{K}_{xu}\mathbf{K}_{uu}^{-1}\mathbf{z}$, but instead of the explicit matrix inversion we utilize a triangular solver for $\bar{\mathbf{f}}_u$ which was numerically more stable. A small diagonal jitter is added to $\mathbf{K}_{uu}$ prior to Cholesky decomposition. The resulting $\mathbf{f}$ is then used directly inside the likelihood during MCMC.

### A.3 Random Fourier Feature Implementation

This subsection details the random Fourier feature (RFF) implementations used in our benchmarks. RFFs approximate a stationary kernel $k_\theta(x - x')$ via a finite dimensional feature map $\phi(x) \in \mathbb{R}^M$ such that

$$k_\theta(x, x') \approx \phi(x)^\top \phi(x'). \tag{20}$$

Following standard constructions [Rahimi and Recht, 2007b], we use cosine features

$$\phi_m(x) = \sqrt{\frac{2}{M}} \cos(\omega_m^\top x + b_m), \tag{21}$$

with phases $b_m \sim \text{Uniform}(0, 2\pi)$ and frequencies $\omega_m$ drawn from the spectral density of the target kernel.

For each experiments (*e.g.*, for each grid size and lengthscale), the base frequencies $\{\omega_m\}_{m=1}^M$ and phases $\{b_m\}_{m=1}^M$ are sampled once and reused throughout inference. The kernel lengthscale $\ell$ is treated as a latent variable and applied by rescaling $\omega_m \mapsto \omega_m/\ell$.

**Kernel-specific sampling.** We use the following standard spectral constructions [Williams and Rasmussen, 2006]:

- **RBF:** $\omega_m \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
- **Matérn-1/2:** $\omega_m = \tan(\pi u_m), \quad u_m \sim \text{Uniform}(-\frac{1}{2}, \frac{1}{2})$.
- **Matérn-3/2:** Frequencies are sampled using the scale-mixture representation of the Matérn-3/2 spectral density. Let $d$ denote the input dimension. Define

  $$\nu = \tfrac{3}{2}, \quad \alpha = \frac{2\nu}{\ell^2} = \frac{3}{\ell^2}, \quad \beta = \nu + \frac{d}{2} = \frac{d+3}{2}.$$

  We sample

  $$t_m \sim \text{Gamma}(\beta, \alpha), \quad \mathbf{z}_m \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d), \quad \omega_m = \frac{\mathbf{z}_m}{\sqrt{2t_m}},$$

  yielding $\omega_m \in \mathbb{R}^d$. Base frequencies are sampled with $\ell = 1$ and rescaled by $\ell^{-1}$ during inference.

During inference, given $\phi(x)$, the latent function is represented as

$$f(x) = \phi(x)^\top \mathbf{z}, \quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{22}$$

yielding a rank-$M$ approximation to the GP prior.

### A.4 Variational Inference Baseline (ADVI)

We use Automatic Differentiation Variational Inference (ADVI) as implemented in NumPyroPhan et al. [2019], with a full-rank Gaussian variational family via the `AutoMultivariateNormal` guide. ADVI is applied to the same probabilistic model used for GP and DeepRV inference, differing only in the posterior approximation.

Let $\mathbf{z}$ denote the collection of all latent variables and hyperparameters in the model. ADVI approximates the posterior $p(\mathbf{z} \mid \mathbf{y})$ with a multivariate Gaussian variational distribution $q_\lambda(\mathbf{z}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\lambda = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$ are variational parameters. These are learned by maximizing the evidence lower bound (ELBO),

$$\mathcal{L}(\lambda) = \mathbb{E}_{q_\lambda(\mathbf{z})}\big[\log p(\mathbf{y}, \mathbf{z}) - \log q_\lambda(\mathbf{z})\big],$$

using reparameterization gradients.

Optimization is performed for a fixed budget of 50,000 iterations using the Adam optimizer with learning rate $10^{-4}$. After optimization, posterior samples are drawn directly from $q_\lambda(\mathbf{z})$ and used to generate posterior predictive samples through the original model likelihood.

### A.5 PriorCVAE workflow

**Algorithm 1** PriorCVAE [Semenova et al., 2023] workflow

Fix the **spatial structure** of interest $\mathbf{s} = (s_1, \ldots, s_n)$, *e.g.* centroids of administrative units
Fix the **latent dimension size** $d \leq n$ for the decoder $D_\psi : \mathbb{R}^d \times \mathcal{C} \to \mathbb{R}^n$, and the encoder $E_\gamma : \mathbb{R}^n \times \mathcal{C} \to \mathbb{R}^d$.

**Train PriorCVAE prior**:
- Sample hyperparameters: $\boldsymbol{\theta} \sim p_{\boldsymbol{\theta}}(\cdot)$.
- Sample GP realizations: $\mathbf{f}_{\boldsymbol{\theta}} \sim \mathcal{GP}_{\boldsymbol{\theta}}(\cdot)$, over the spatial structure $\mathbf{s}$
- Encode $\hat{\mathbf{z}}_\mu, \hat{\mathbf{z}}_\sigma = E_\gamma(\mathbf{f}_{\boldsymbol{\theta}}, \boldsymbol{\theta})$, sample $\hat{\mathbf{z}} \sim \mathcal{N}(\hat{\mathbf{z}}_\mu, \hat{\mathbf{z}}_\sigma)$, and decode $\hat{\mathbf{f}}_{\boldsymbol{\theta}} = D_\psi(\hat{\mathbf{z}}, \boldsymbol{\theta})$.
- Back propagate the loss:$\mathcal{L}_{\text{CVAE}} = \frac{1}{\sigma^2_{\text{vae}}} \text{MSE}(\mathbf{f}_{\boldsymbol{\theta}}, \hat{\mathbf{f}}_{\boldsymbol{\theta}}) + \text{KL}\left[\mathcal{N}(\hat{\mathbf{z}}_\mu, \hat{\mathbf{z}}_\sigma) || \mathcal{N}(\mathbf{0}, \mathbf{1})\right]$

**Perform Bayesian inference with MCMC** of the overarching model, including latent variables and hyperparameters $\boldsymbol{\theta}$, by approximating $f_{\boldsymbol{\theta}}$ with $\hat{\mathbf{f}}_{\boldsymbol{\theta}}$ in a drop-in manner using the trained decoder:

$$\mathbf{f}_{\boldsymbol{\theta}} \approx \hat{\mathbf{f}}_{\boldsymbol{\theta}} = D_\psi(\mathbf{z}, \boldsymbol{\theta}), \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$$

# B Experiments

## B.1 Benchmarking DeepRV

### B.1.1 Experimental details

**Models and architectures.** DeepRV variants included a two-layer MLP with ReLU activations, a two-layer gMLP without bottleneck, and a transformer with kernel-based attention bias. PriorCVAE used a standard MLP encoder–decoder. Inducing points and RFFs were implemented in NumPyro. INLA was run with the `R-INLA` package.

**Training setup.** DeepRV and PriorCVAE were trained with batch size 32 using Optax optimizers with cosine-annealed learning rates and gradient clipping ($\|\cdot\|_2 \leq 3$). DeepRV–gMLP used AdamW with maximum learning rate $10^{-3}$ ($N \leq 32^2$) or $2 \times 10^{-3}$ for larger grids. DeepRV–MLP and PriorCVAE used learning rates of $10^{-3}$ for small grids and up to $5 \times 10^{-3}$ otherwise. Training ran for 200,000 steps (300,000 steps for $48^2$ and $64^2$ grids). ADVI optimization was performed with Adam at a fixed learning rate of $10^{-4}$ for 50,000 steps.

**Priors.** For the Matérn-1/2 kernel, the lengthscale prior was $\ell \sim \text{LogNormal}(3.0, 0.4)$, the variance was fixed at 1, and $\beta \sim \mathcal{N}(0, 1000)$. For the Matérn-3/2 kernel, the prior was $\ell \sim \text{LogScaleTransform}(\text{Beta}(4, 1))$ spanning $(1, 100)$, the variance was fixed at 1, and $\beta \sim \mathcal{N}(0, 1)$.

**Hardware.** All Matérn-1/2 experiments were run on a single NVIDIA GeForce RTX 5090 GPU. Matérn-3/2 experiments used an NVIDIA RTX 5000 Ada GPU. INLA computations were performed on a Mac CPU.

**Training times.** Average training times (in seconds) for DeepRV and PriorCVAE across grid sizes are reported in Table 6. Each entry shows the mean ± standard error over three runs.

| Metric | Grid | DeepRV-MLP | DeepRV-gMLP | PriorCVAE |
|---|---|---|---|---|
| Train Time (s) | 256 | 163.33 ± 1.24 | 247.29 ± 1.60 | 183.26 ± 1.51 |
| | 576 | 165.71 ± 1.43 | 249.85 ± 1.44 | 185.69 ± 2.57 |
| | 1024 | 165.86 ± 1.20 | 360.01 ± 0.17 | 185.97 ± 1.29 |
| | 2304 | 448.37 ± 0.23 | 1297.65 ± 0.32 | 632.58 ± 0.39 |
| | 4096 | 1895.87 ± 2.78 | 4106.42 ± 12.66 | 2999.56 ± 2.94 |

Table 6: Training times (in seconds) for DeepRV and PriorCVAE across grid sizes for the Matérn-1/2 kernel, averaged over three runs with standard error.

### B.1.2 Results: Matérn-3/2

We repeated the benchmarking experiment using the Matérn-3/2 kernel, which is not directly supported by standard `R-INLA`. Details of the setup follow Section 6.1, with the modified prior described above. Observations were generated from the Poisson model in Eq. 9, and inference was performed with NUTS [Hoffman and Gelman, 2014] (4 chains for $32^2$, one chain otherwise; 4,000 warmup steps and 10,000 posterior samples). True lengthscales $\ell \in \{10, 30, 50\}$ and $\beta = 1$ were used, with ∼50% of observations masked in spatially contiguous regions. Results presented in Table 7,Figure 6 and are consistent with the Matérn-1/2 results. Here Inducing Points are able to approximate the GP better as the kernel is smoother.

## B.2 DeepRV flexibility: non-separable spatiotemporal kernel

### B.2.1 Experimental details

**Models and architectures.** We used a two-layer gMLP variant of DeepRV. PriorCVAE employed a standard MLP encoder–decoder. Inducing points, ADVI, and baseline GP were also implemented for comparison.

**Training setup.** DeepRV and PriorCVAE were trained with batch size 32 for 500,000 steps. Training used the same optimizers as in the benchmarking experiment: AdamW (cosine-annealed learning rate, gradient clipping $\|\cdot\|_2 \leq 3$) for DeepRV, and Yogi for PriorCVAE. ADVI optimization was performed with Adam at a fixed learning rate of $10^{-4}$ for 50,000 steps.

| metric | Grid | ADVI | GP | DeepRV-MLP | DeepRV-gMLP | Inducing Pts | PriorCVAE | RFF |
|---|---|---|---|---|---|---|---|---|
| MSE($\hat{\mathbf{y}}_{gp}, \hat{\mathbf{y}}$) | 256 | 0.667 ± 0.38 | - | 1.216 ± 1.17 | 0.002 ± 0.00 | 0.380 ± 0.35 | 106.077 ± 105.60 | 0.223 ± 0.13 |
| | 576 | 1.183 ± 0.55 | - | 1.213 ± 0.91 | 0.006 ± 0.00 | 0.397 ± 0.24 | 3.383 ± 1.63 | 1.655 ± 1.03 |
| | 1024 | 1.463 ± 0.61 | - | 0.853 ± 0.76 | 0.014 ± 0.01 | 0.804 ± 0.78 | 5.704 ± 5.41 | 12.939 ± 12.84 |
| | 2304 | 1.479 ± 0.37 | - | 0.165 ± 0.13 | 0.002 ± 0.00 | 0.152 ± 0.14 | 1.445 ± 1.30 | 0.958 ± 0.91 |
| | 4096 | 5.755 ± 1.76 | - | 1.000 ± 0.35 | 0.024 ± 0.01 | 0.334 ± 0.29 | 1.395 ± 1.02 | 0.740 ± 0.36 |
| Wass($\hat{\ell}_{gp}, \hat{\ell}$) | 256 | 26.94 ± 15.51 | - | 2.92 ± 1.08 | 0.31 ± 0.14 | 0.66 ± 0.40 | 24.07 ± 2.63 | 14.22 ± 7.70 |
| | 576 | 15.19 ± 8.15 | - | 4.04 ± 1.67 | 0.20 ± 0.09 | 0.75 ± 0.15 | 13.66 ± 5.22 | 16.74 ± 3.71 |
| | 1024 | 19.40 ± 11.12 | - | 3.93 ± 1.11 | 0.23 ± 0.13 | 0.49 ± 0.22 | 14.09 ± 5.80 | 28.95 ± 13.22 |
| | 2304 | 26.44 ± 12.27 | - | 1.79 ± 0.47 | 0.19 ± 0.08 | 0.32 ± 0.12 | 13.37 ± 3.51 | 14.47 ± 3.74 |
| | 4096 | 24.25 ± 11.27 | - | 1.29 ± 0.14 | 0.22 ± 0.07 | 0.19 ± 0.05 | 18.92 ± 5.62 | 6.97 ± 2.82 |
| ESS ($\ell$)/sec | 256 | 326.11 ± 5.45 | 12.66 ± 6.27 | 17.31 ± 9.25 | 18.95 ± 8.49 | 20.68 ± 12.62 | 37.98 ± 20.08 | 2.04 ± 1.06 |
| | 576 | 199.35 ± 1.04 | 5.54 ± 2.22 | 9.91 ± 5.58 | 14.05 ± 5.29 | 12.25 ± 4.30 | 18.19 ± 13.45 | 15.66 ± 14.95 |
| | 1024 | 483.24 ± 19.58 | 2.40 ± 0.67 | 10.24 ± 1.87 | 10.12 ± 2.22 | 8.27 ± 2.24 | 21.45 ± 11.64 | 0.44 ± 0.27 |
| | 2304 | 33.06 ± 0.63 | 0.33 ± 0.11 | 1.94 ± 0.44 | 4.79 ± 1.11 | 2.98 ± 0.62 | 4.94 ± 2.41 | 2.13 ± 1.92 |
| | 4096 | 8.79 ± 0.04 | 0.05 ± 0.03 | 0.65 ± 0.18 | 1.24 ± 0.53 | 2.38 ± 0.97 | 1.29 ± 0.66 | 0.13 ± 0.12 |
| Infer Time (s) | 256 | 8 ± 0.13 | 364 ± 109.52 | 138 ± 48.00 | 188 ± 43.79 | 433 ± 269.17 | 100 ± 15.04 | 219 ± 16.85 |
| | 576 | 12 ± 0.08 | 1294 ± 466.29 | 230 ± 45.33 | 381 ± 87.95 | 391 ± 117.26 | 213 ± 32.01 | 304 ± 9.72 |
| | 1024 | 21 ± 0.80 | 2686 ± 781.70 | 350 ± 60.23 | 583 ± 131.98 | 456 ± 111.84 | 304 ± 69.96 | 497 ± 63.00 |
| | 2304 | 75 ± 1.46 | 14197 ± 2781.03 | 1176 ± 106.86 | 984 ± 178.77 | 892 ± 119.61 | 946 ± 27.88 | 1606 ± 4.14 |
| | 4096 | 285 ± 0.38 | 139912 ± 64389.16 | 3950 ± 969.34 | 5923 ± 3301.25 | 1792 ± 745.92 | 2920 ± 657.59 | 7273 ± 2625.99 |

Table 7: Matérn-3/2 benchmarking results: (a) Posterior predictive MSE relative to full GP MCMC; (b) Wasserstein distance between inferred and full GP MCMC lengthscale posteriors; (c) Effective $\ell$ sample size (ESS) per second; (d) Inference time in seconds. Results are shown for each grid size and are averaged across the three true lengthscales (10, 30, 50) over 15 runs, with the standard error reported.

**Priors.** For inference we used:

$$\ell \sim \text{LogScaleTransform}(\text{Beta}(4,1)),\ a \sim \text{LogNormal}(0,1),\ \alpha \sim \text{Beta}(2,2),\ \nu \sim \text{Uniform}(D, 2D),\ \beta \sim \mathcal{N}(0,1),$$

with variance fixed at 1.0. Data were generated with hyperparameters $\ell = 20.0$, $\beta = 1.0$, $a = 0.5$, $\alpha = 0.8$, $b = 1.0$, $\nu = 1.0$, and a dimensionality of $D = 2$.

**Hardware.** Experiments were run on a single NVIDIA GeForce RTX 5090 GPU, consistent with the Matérn-1/2 benchmarks.

**Training times.** Training times (in seconds) are shown in Table 8. Each entry is the mean ± standard error across three runs.

| Model | Train time (s) | Infer time (s) |
|---|---|---|
| Baseline GP | – | 5751.35 |
| Inducing Points | – | 965.91 |
| PriorCVAE | 837.38 | 986.12 |
| ADVI | – | 105.48 |
| DeepRV-gMLP | 1164.62 | 1099.13 |

Table 8: Training and inference times (in seconds) for the non-separable spatiotemporal kernel. Train times are reported where models required pre-training. Inference times are reported for all models.

### B.2.2 Results

The resulting posterior predictive of the top models are shown in Figure 7. DeepRV closely tracks the GP baseline across space and time, while inducing points and PriorCVAE exhibit higher deviations.

## B.3 Real-world application: London LSOA dataset

### B.3.1 Experimental details

**Models and architectures.** We used a two-layer gMLP variant of DeepRV. No other approximations (*e.g.*, inducing points, PriorCVAE) were benchmarked in this experiment; comparisons were made only against the GP baseline.

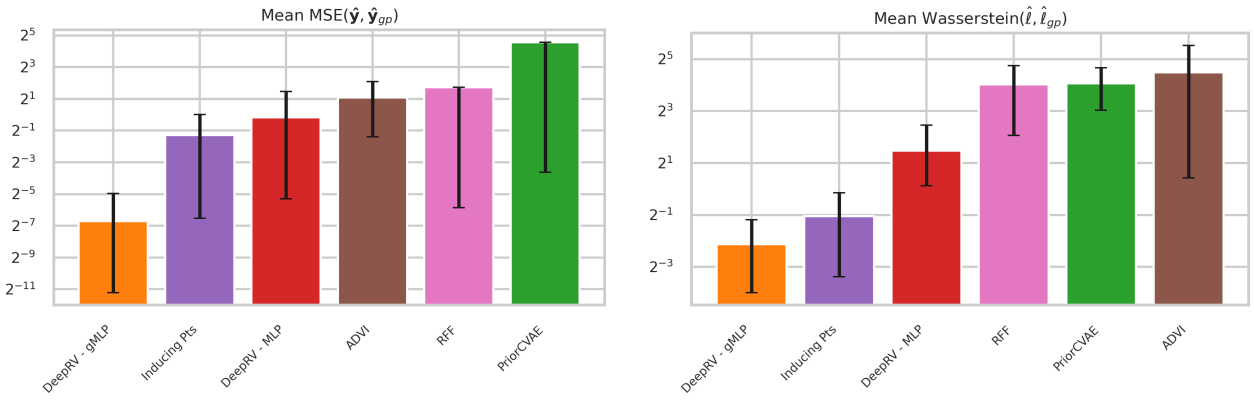


Figure 6: (a) Posterior predictive MSE relative to full GP MCMC; (b) Wasserstein distance between inferred and full GP MCMC lengthscale posteriors. Y axis is $\log_2$-scaled, to allow a clear comparison of all methods. Results are averaged across true lengthscales and grid sizes over 15 runs, with 10% and 90% quantiles reported.

**Training setup.** DeepRV was trained with batch size 16 for 500,000 steps using the AdamW optimizer with cosine-annealed learning rate schedule and gradient clipping ($\|\cdot\|_2 \leq 3$).

**Priors.** For training we used $\ell \sim \text{Uniform}(1.0, 55.0)$, with variance fixed at 1.0. We slightly tightened the usual prior from Uniform$(1.0, 55.0)$, as the MAP values of lengthscale were relatively small. Thus, this prior supports lengthscale between $0-50$, and is expansive enough for this dataset. For inference, priors were centered at MAP values from initialization:

$$\sigma^2 \sim \text{LogNormal}(\log(\max(\text{var}_{\text{MAP}}, 10^{-3})), 0.75), \quad \ell \sim \text{Gamma}(4, 4/\ell_{\text{MAP}}), \quad \beta \sim \mathcal{N}(\beta_{\text{MAP}}, 1.0).$$

**Hardware.** Experiments were run on a single NVIDIA RTX 5000 Ada GPU, consistent with the Matérn-3/2 benchmarks.

**Training and inference times.** Training required approximately 12,885 seconds (∼3.6 hours) at the LSOA level and 1,371 seconds (∼23 minutes) at the MSOA level. Inference required 9,081 seconds at LSOA and 3,009 seconds at MSOA. For validation, the MSOA full GP was run with 4 chains of 4,000 warmup and 4,000 posterior samples, while the LSOA short GP calibration run used 2 chains with 1,000 warmup and 500 posterior samples.

### B.3.2 Results

Observed versus predicted prevalence comparisons for unobserved locations are shown in Figures 8 and 9. Model-vs-model comparisons of DeepRV against the GP baseline are shown in Figure 10. These results confirm that DeepRV produces predictions and uncertainty estimates closely aligned with the GP baseline at both MSOA and LSOA levels.

## B.4 Arbitrary-Locations DeepRV

### B.4.1 Experimental details

**Models and architectures.** We used a Transformer-based DeepRV with four layers, embedding dimension $D = 128$, four attention heads, kernel-attention bias, and identity embeddings. This architecture allows the model to handle arbitrary sets of locations up to a maximum specified at training time. Baselines included the full GP and inducing points.

**Training setup.** DeepRV was trained with batch size 8 for 2M steps using the AdamW optimizer with learning rate $10^{-4}$, cosine annealing, and gradient clipping ($\|\cdot\|_2 \leq 3$). Inducing points were trained with 600k steps. GP required no pre-training. Additionally, to validate the claim that the transformer DeepRV variant could not learn without ID embeddings we provide ablation results for this model with: (a) only RFF positional embeddings without ID embeddings, and (b) only fixed Sinusoidal positional embeddings without ID embeddings. Results for this ablation test are reported in Table 10.

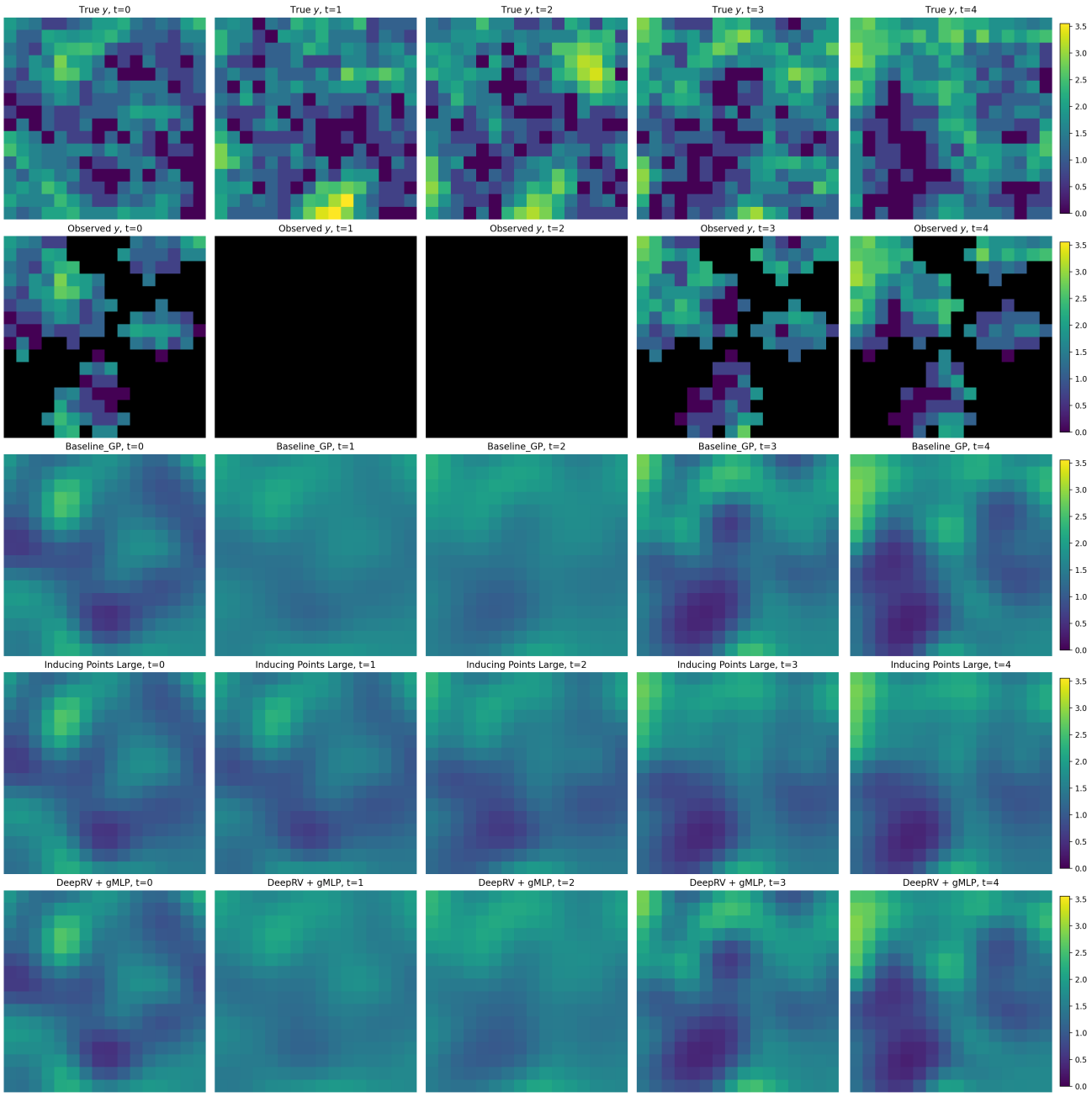


Figure 7: Non-separable spatiotemporal kernel posterior predictives. Results for the top models are presented across time steps.

**Priors.** For both training and inference, we used $\ell \sim \text{Uniform}(1.0, 50.0)$ with variance fixed at 1. The data were generated with true hyperparameters $\ell = 20.0$, $\beta = 1.0$, $\sigma^2 = 1.0$. Inference priors for DeepRV, GP, and Inducing were identical.

**Hardware.** Experiments were run on a single NVIDIA RTX 5090 GPU, consistent with the Matérn-1/2 benchmarks.

**Training and inference times.** DeepRV was trained once, requiring ∼43,426 seconds (≈12 hours). Inference used two chains with 2,000 warmup and 4,000 posterior samples. Table 9 reports training and inference times.

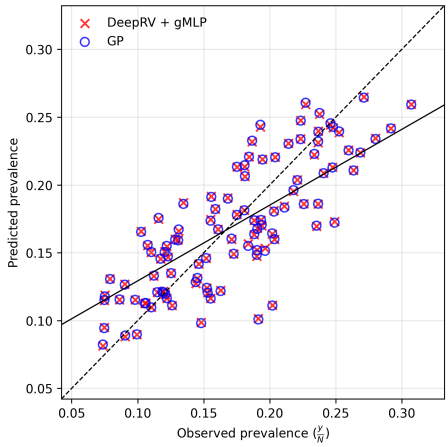


Figure 8: Observed versus predicted prevalence ($\mathbf{p}$ in Equation 10) at 100 randomly selected unobserved MSOA locations. Each point represents one MSOA. The black full line shows the linear regression of DeepRV predictions, illustrating the smoothing effect of the model while maintaining fidelity to the full GP MCMC's prevalence.

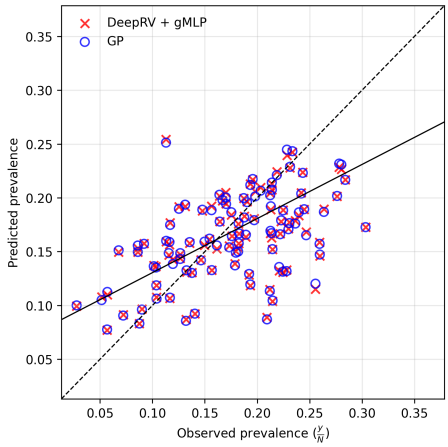


Figure 9: Observed versus predicted prevalence ($\mathbf{p}$ in Equation 10) at 100 randomly selected unobserved LSOA locations. Each point represents one LSOA. The black full line shows the linear regression of DeepRV predictions, illustrating the smoothing effect of the model while maintaining fidelity to the full GP MCMC's prevalence.

| Model | Train time (s) | Infer time (s) $N = 512$ | Infer time (s) $N = 1024$ | Infer time (s) $N = 2048$ |
|---|---|---|---|---|
| GP | – | 644.31 | 2903.27 | 7729.96 |
| Inducing Pts | – | 192.60 | 556.58 | 825.74 |
| DeepRV | 43425.74 | 512.30 | 2680.88 | 7197.59 |

Table 9: Training and inference times (in seconds) for the arbitrary-locations experiment. Inference times are reported per grid size ($N = 512, 1024, 2048$). DeepRV was trained once and then applied to all datasets, while GP and Inducing Points require no pre-training.

### B.4.2 Results

Posterior distribution comparisons across dataset sizes are shown in Figure 11. DeepRV closely matches GP posteriors, while inducing points show larger deviations. The ablation results in Table 10 show that models without ID embeddings fail to learn any structure, collapsing to near-zero function realisations. This produces an artefact of low predictive MSE, despite incorrect inference, as the MCMC overfits the observed data by adjusting $\beta$ and latent $\mathbf{z}$, yielding posterior predictive means similar to the GP baseline while severely misestimating the

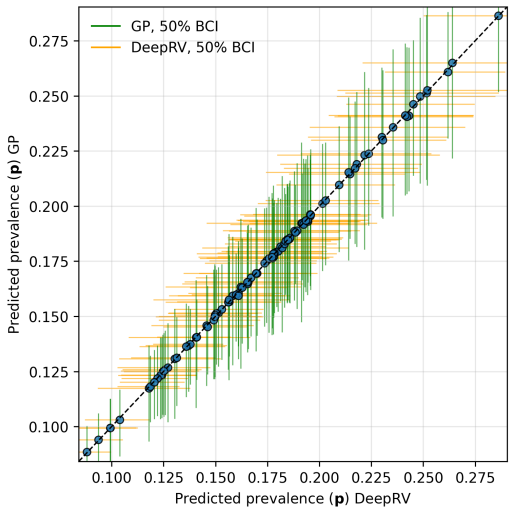


Figure 10: Predicted prevalence ($\mathbf{p}$ in Equation 10) from DeepRV compared against the full GP baseline at 100 randomly selected locations. Each point represents one LSOA. Vertical and horizontal lines denote 50% credible intervals of models.

| Model | Test Loss | MSE($\hat{\mathbf{y}}_{gp}, \hat{\mathbf{y}}$) | Wass($\hat{\ell}_{gp}, \hat{\ell}$) | LPD | Cover-80% |
|---|---|---|---|---|---|
| GP | - | - | - | **-2.00 ± 0.08** | **0.97 ± 0.01** |
| DeepRV | **0.007** | **0.01 ± 0.01** | **0.66 ± 0.20** | **-2.00 ± 0.08** | **0.97 ± 0.01** |
| DeepRV RFF only | 0.823 | 0.06 ± 0.04 | 10.07 ± 0.15 | -1.99 ± 0.14 | **0.98 ± 0.01** |
| DeepRV Sinusoidal only | 0.823 | 0.07 ± 0.05 | 10.16 ± 0.15 | -1.98 ± 0.14 | **0.98 ± 0.00** |
| Inducing Points | - | 1.82 ± 1.10 | 3.88 ± 0.15 | -2.09 ± 0.10 | 0.86 ± 0.02 |

Table 10: Full Arbitrary-locations experiment results including positional encoding ablations: (a) Posterior predictive MSE relative to GP; (b) Wasserstein distance between inferred and GP lengthscale posteriors; (c) Log predictive density (LPD); (d) Coverage of the 80% posterior predictive. Results are averaged across dataset sizes, with the standard error reported.

lengthscale posterior, and failing to infer the underlaying process.

### B.5 Ablation Study

#### B.5.1 Experimental details

**Models and architectures.** We compared a two-layer DeepRV–MLP with ReLU activations, a two-layer DeepRV–gMLP, and a two-layer DeepRV–Transformer with kernel attention and identity embeddings. PriorCVAE used a standard two-layer MLP encoder–decoder. The full GP baseline was also included for comparison.

**Training setup.** All models were trained with batch size 32 for 200,000 steps. Optimizers followed the benchmarking setup: AdamW with cosine-annealed learning rate schedule and gradient clipping ($\|\cdot\|_2 \leq 3$) for all DeepRV models except DeepRV–MLP, which used Adam; PriorCVAE used the Yogi optimizer.

**Priors.** For both training and inference, the lengthscale prior was uniform across the grid $(0, 100)$ and $\beta \sim \mathcal{N}(0, 1)$. Data were generated from four kernels (Matérn–1/2, Matérn–3/2, Matérn–5/2, and RBF) with hyperparameters drawn from

$$\ell \sim \text{Uniform}(5, 50), \quad \beta \sim \text{Uniform}(0.6, 2.0),$$

with three seeds per kernel.

**Hardware.** All experiments were run on a single NVIDIA RTX 5000 Ada GPU, consistent with the Matérn–3/2 benchmarks.

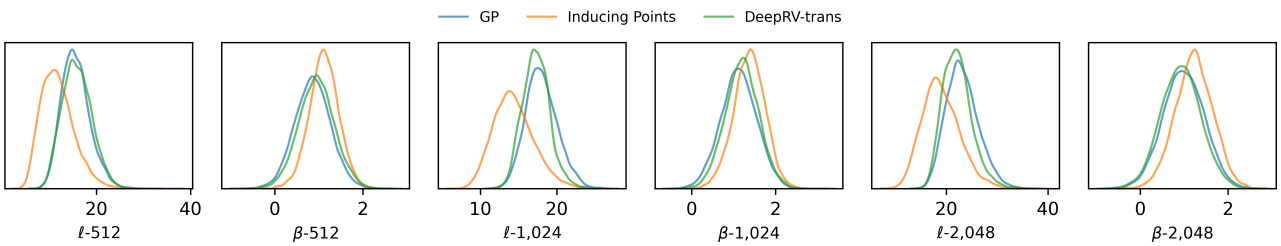


Figure 11: Arbitrary-locations inferred hyperparameter posterior distributions per dataset size ($N = 512, 1024, 2048$). DeepRV closely matches GP posteriors across scales, while inducing points deviate.

**Training and inference times.** Training and inference times were averaged across kernels and seeds. Reported values are provided in Table 5 in the main text.

### B.5.2 Results

Full ablation results, averaged across kernels and seeds, are reported in the main text (Table 4). In summary, the gap between PriorCVAE and DeepRV–MLP stems from the decoder-only design, while the Transformer achieves accuracy close to gMLP at substantially higher computational cost. All DeepRV variants closely track the GP baseline in both predictive and parameter inference, with gMLP offering the best balance of accuracy and efficiency. Additionally, we measured the cross-chain accuracy of DeepRV to validate that it remains accurate even when not starting from the same seed as the GP. The results were consistent with the main Table 4, with the cross chain $\text{MSE}(\hat{\mathbf{y}}, \mathbf{y}_{\text{gp}})$ of $0.012 \pm 0.016$, and $\text{Wasserstein}(\hat{\ell}, \ell_{\text{gp}})$ of $0.272 \pm 0.243$.

## B.6 Vecchia Comparison

### B.6.1 Experimental details

**Models and architectures.** We compare three samplers: (i) the full GP process, (ii) a Vecchia approximation, and (iii) a learned gMLP-based DeepRV surrogate.

**Spatial designs and kernels.** We examine the approximations on four kernels, RBF, Matérn-1/2, Matérn-3/2, and Matérn-5/2, for each kernel we generate five independent spatial locations by sampling $N = 2304$ locations uniformly from $[0, 100]^2$, enforcing a minimal separation between points. For each spatial design, we draw 1,000 independent samples from the GP prior and from the corresponding Vecchia and DeepRV approximations, by sampling lengthscale uniformaly from $0 - 100$, and fixing variance to 1. In total we compare the methods on $20,000$ different samples.

**Vecchia setup.** The Vecchia approximation factorises the joint GP prior as

$$p(\mathbf{f}) \approx \prod_{i=1}^{N} p\big(f_i \mid \mathbf{f}_{C(i)}\big),$$

where each conditioning set $C(i)$ contains at most $M$ previously ordered locations. For each $i$, the conditional distribution is Gaussian with

$$f_i = \mathbf{a}_i^\top \mathbf{f}_{C(i)} + \sqrt{v_i}\, z_i, \qquad \mathbf{a}_i = K_{C(i),C(i)}^{-1} K_{C(i),i}, \qquad v_i = K_{ii} - K_{i,C(i)} K_{C(i),C(i)}^{-1} K_{C(i),i}.$$

We set the conditioning size to $M = 2\sqrt{N}$, matching the computational complexity of DeepRV. Conditioning sets are selected using nearest neighbours in input space, and no specialised ordering strategies are employed. We further enhanced the sampling speed of this approach by parallelising the computation of $\mathbf{a}$ and $\mathbf{v}$, and using `jax.lax.scan` to compute $\mathbf{f}$ faster.

**Training setup (DeepRV).** DeepRV is trained once per spatial design and kernel using batch size 32 for 300,000 optimisation steps with the AdamW optimizer, cosine-annealed learning rate, and gradient clipping ($\|\cdot\|_2 \leq 3$). We note that one training run of the RBF kernel did not converge (validation loss $> 0.01$), and is repeated with a different seed, the resampling of the seed is done automatically, and is integrated in the code.

**Hardware.** All experiments were run on a single NVIDIA RTX 5090 GPU.

### B.6.2 Results

| Kernel | GP time (s) | Vecchia time (s) | DeepRV time (s) | Vecchia MSE | DeepRV MSE |
|---|---|---|---|---|---|
| Matérn-1/2 | $0.0011 \pm 0.0002$ | $0.0148 \pm 0.0001$ | $\mathbf{0.0002 \pm 0.0000}$ | $\mathbf{0.0000 \pm 0.0000}$ | $\mathbf{0.0000 \pm 0.0002}$ |
| Matérn-3/2 | $0.0010 \pm 0.0002$ | $0.0150 \pm 0.0001$ | $\mathbf{0.0002 \pm 0.0000}$ | $\mathbf{0.0002 \pm 0.0001}$ | $\mathbf{0.0002 \pm 0.0006}$ |
| Matérn-5/2 | $0.0011 \pm 0.0002$ | $0.0150 \pm 0.0001$ | $\mathbf{0.0002 \pm 0.0000}$ | $\mathbf{0.0004 \pm 0.0003}$ | $0.0008 \pm 0.0022$ |
| RBF | $0.0011 \pm 0.0002$ | $0.0150 \pm 0.0001$ | $\mathbf{0.0002 \pm 0.0000}$ | $\mathbf{0.0007 \pm 0.0003}$ | $0.0016 \pm 0.0118$ |

Table 11: Mean ± standard deviation of sampling time and absolute MSE with respect to a full GP. Results are averaged over 5 spatial locations and 1,000 samples per location set.

As shown in Table 11, Vecchia attains very high accuracy relative to the exact GP, in some cases even outperforming DeepRV on the smoother Matérn–5/2 and RBF kernels. However, despite additional implementation-level optimizations, the average time required to generate a single Vecchia sample exceeds that of the dense GP and is approximately two orders of magnitude slower than DeepRV. This behavior is likely due to the fact that dense GP sampling can be fully parallelized on a GPU, whereas Vecchia relies on an inherently sequential computation over conditioning sets, which becomes a bottleneck at this scale. We therefore decided that these runtime costs would make Vecchia comparisons infeasible for the main paper's benchmarking.