DEEPRV: PRE-TRAINED SPATIAL PRIORS FOR ACCELERATED DISEASE MAPPING.

Anonymous authors
Paper under double-blind review

ABSTRACT

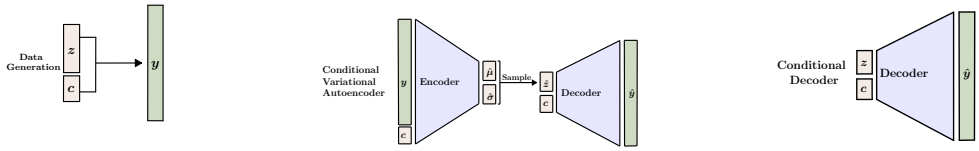
Recently introduced deep generative priors (*e.g.*, PriorVAE, πVAE, and PriorCVAE) have emerged as powerful tools for scalable Bayesian inference by emulating complex stochastic processes like Gaussian processes (GPs). However, these methods remain largely a proof-of-concept and inaccessible to practitioners. We propose **DeepRV**, a lightweight, decoder-only approach that reduces the number of parameters by 66%, accelerates training, and enhances real-world applicability in comparison to current VAE-based approaches. Leveraging probabilistic programming frameworks (*e.g.*, NumPyro), DeepRV achieves an order-of-magnitude speedup while maintaining robust performance. We showcase its effectiveness in GP emulation and spatial analysis of the UK using simulated data and cancer mortality rates (**Work in progress**). To bridge the gap between theory and practice, we provide a user-friendly API, enabling scalable and efficient Bayesian inference.

1 INTRODUCTION

Bayesian modeling serves as a fundamental framework analyzing spatial data, particularly in disease mapping, where Gaussian processes (GPs) (Williams & Rasmussen, 2006) and its close relatives, such as the conditional auto-regressive (CAR, Besag (1974)), the Intrinsic Conditional Auto-Regressive (ICAR, Besag & Kooperberg (1995)), and Besag-Yorg-Mollie (BYM, Besag et al. (1991)) models are frequently employed to capture spatial dependencies. However, sampling from these models incurs a computational complexity of $\mathcal{O}(n^3)$, as inverting the covariance matrix is required, posing scalability challenges for large datasets. Recent advances in deep generative models (DGMs) have introduced the concept of pre-training priors for Bayesian inference, offering a potential solution to these bottlenecks. Frameworks such as PriorVAE (Semenova et al., 2022), πVAE (Mishra et al., 2022), PriorCVAE (Semenova et al., 2023b), and aggVAE (Semenova et al., 2023a) have demonstrated the feasibility of encoding complex priors using DGMs, enabling faster inference by decoupling the computational burden of prior construction from the inference stage.

However, these methods remain largely a proof-of-concept and inaccessible to practitioners. They often require substantial expertise in deep learning model implementation, and their computational efficiency gains are offset by the complexity of training variational autoencoders (VAEs) (Kingma, 2013). Further, VAEs are prone to *posterior collapse*, a phenomenon where the generative model effectively disregards a subset of the latent variables, thereby failing to capture meaningful variations in the data (Lucas et al., 2019; Wang et al., 2021). Additionally, they can suffer from *oversmoothing* induced by the latent bottleneck, where the reconstructed outputs lack sharpness and fine-details (Takida et al., 2022). Moreover, we show in this paper that when emulating GPs, the most commonly used spatial priors, the encoder-decoder architecture introduces unnecessary overhead for applications focused solely on inference. Consequently, their adoption in applied fields like disease mapping remains limited, where ease of use, efficiency, and reliability are crucial.

In practice, disease mapping workflows often rely either on fast, black-box tools such as R-INLA (Lindgren & Rue, 2015), or slow, bespoke models using Probabilistic Programming Languages (PPLs) (Carpenter et al., 2017; Abril-Pla et al., 2023; Phan et al., 2019; Bingham et al., 2019; de Valpine et al., 2017). While PPLs offer flexibility and robust inference via Markov Chain Monte Carlo (MCMC), their performance and computational demands suffer when modeling large spatial datasets, particularly when using Gaussian processes. This trade-off between computational



(a) Data generation process (b) PriorCVAE architecture (c) DeepRV architecture

Figure 1: (a) Data generation process; (b) PriorCVAE architecture: both the encoder and decoder get the label c as input. Here we interpret c as a hyperparameter of the prior; (c) deepRV architecture.

efficiency and model flexibility creates a need for novel solutions that integrate the benefits of pre-trained priors with practical usability for spatial modeling.

To address these challenges, we introduce **DeepRV**, a novel training procedure and architecture that simplifies spatial modeling. Rather than using realizations of the stochastic process as training data for pre-trained priors, our approach utilizes the latent space as training input, enabling DeepRV to directly map the latent variables to the prior space. This eliminates the need for an encoder, reducing oversmoothing, lowering the parameter count, simplifying training, and accelerating inference while preserving prior richness. Our contributions are threefold:

- *The novel* **DeepRV** *architecture and training process*: A decoder-only generative model that improves reconstruction and accelerates training compared to VAE-based methods.
- *UK case-studies*: we demonstrate the efficacy of DeepRV on both simulated data for UK Lower Tier Local Authorities (LTLA) and real-world 2023 cancer mortality data for UK Local Administrative Districts (LADs) **(Work in progress)**, showing substantial speedups and robust inference performance as compared to directly sampling GPs in Numpyro.
- *API for practical usability*: we provide code for generating pre-trained priors, bridging the gap between methodological advances in Bayesian modeling and their practical application in disease mapping.

Our results show that inference with DeepRV in Numpyro, as opposed to directly sampling GPs, is an order of magnitude faster, making it a scalable tool for disease mapping. By addressing key computational and usability challenges, DeepRV lays the foundation for broader adoption of pre-trained priors in Bayesian inference workflows.

2 VAE-BASED ENCODING PRIORS

PriorVAE (Semenova et al., 2022) and πVAE (Mishra et al., 2022) are two related VAE-based methods, with PriorVAE encoding finite realizations of stochastic processes and πVAE capturing their continuous representations. They use a two-stage approach where a VAE decoder, pre-trained to approximate complex priors, is used for sampling within Bayesian inference with MCMC – preserving MCMC rigor while enhancing scalability. πVAE learns low-dimensional embeddings of function classes inspired by the Karhunen–Loéve expansion, where basis functions are learned via a neural network, and random coefficients are encoded with a VAE. PriorVAE, in contrast, follows a standard VAE framework to encode finite realizations of a stochastic prior in question, such as GPs realizations, commonly used in spatial statistics. A key limitation of π- and PriorVAE is their inability to encode and infer hyperparameters of the stochastic process. PriorCVAE (Semenova et al., 2023b) addresses this by introducing a conditional VAE (CVAE) architecture (Sohn et al., 2015), allowing priors to be conditioned on parameters of interest by incorporating them into both encoder and decoder inputs. This enhances prior flexibility and enables parameter inference. The PriorCVAE workflow is presented in the Appendix, Algorithm 2. However, PriorCVAE still suffers from VAE limitations such as posterior collapse and oversmoothing, limiting practical usability. In this work, we build on PriorCVAE to develop a simplified approach for conditioning and sampling pre-trained stochastic process priors.

3 THE PROPOSED METHOD: DEEPRV

In this section, we introduce **DeepRV**, a decoder-only deep generative model designed for efficiently approximate priors for inference. Unlike VAEs, which map inputs to a latent space and reconstruct samples via a learned decoder, DeepRV directly maps a fixed latent distribution to realizations of stochastic processes. The design of PriorCVAE and DeepRV are illustrated in Figure 1, highlighting their shared data generation process (Figure 1, a), and their structural differences (Figure 1, b-c).

3.1 TRAINING AND INFERENCE WORKFLOW

The first step of the DeepRV workflow approximates realizations of the stochastic process $F_{\mathbf{c}}(\cdot)$, conditioned on hyperparameters $\mathbf{c} \sim p_{\mathcal{C}}(\cdot)$, at a fixed set of n locations $\mathbf{x} = (x_1, \ldots, x_n)^\top$ within the index set. By $\mathbf{f_c} = F_{\mathbf{c}}(\mathbf{x})$, we denote the vector of realizations of the process at the fixed locations, which forms a *random variable* – hence the "RV" abbreviation in the method's name. To model these realizations, a latent space $\mathcal{Z} \subseteq \mathbb{R}^n$ is introduced, along with a probability distribution $\mathcal{D}_{\mathcal{Z}}(\cdot)$ over this space and a deterministic mapping $T_{\mathbf{c}}$. The map $T_{\mathbf{c}}$ is such that, when applied to latent samples $\mathbf{z} \sim \mathcal{D}_{\mathcal{Z}}(\cdot)$, it produces outputs distributed as the stochastic process realizations $\mathbf{f_c}$. We train a deep neural network – hence the "Deep" abbreviation in the method's name – to approximate $T_{\mathbf{c}}$ by minimizing the reconstruction loss between the true process realization $\mathbf{f_c}$ and the model-generated output $\hat{\mathbf{f}}_{\mathbf{c}}$:

$$\mathbf{c} \sim p_{\mathcal{C}}(\cdot), \quad \mathbf{z} \sim \mathcal{D}_{\mathcal{Z}}(\cdot),$$
$$\mathbf{f_c} = T_{\mathbf{c}}(\mathbf{z}), \quad \hat{\mathbf{f}}_{\mathbf{c}} = \mathrm{DeepRV}(\mathbf{z}, \mathbf{c}),$$
$$\mathcal{L}_{\text{DeepRV}} = \mathrm{MSE}(\mathbf{f_c}, \hat{\mathbf{f}}_{\mathbf{c}}).$$

After training, the inference process involves replacing the realizations of the stochastic process $F_{\mathbf{c}}$, $\mathbf{f_c}$, by sampling from the latent distribution $\mathcal{D}_{\mathcal{Z}}(\cdot)$, the hyperparameter priors $p_{\mathcal{C}}(\cdot)$, and feeding them into the trained DeepRV network to generate posterior samples:

$$\mathbf{c} \sim p_{\mathcal{C}}(\cdot), \quad \mathbf{z} \sim \mathcal{D}_{\mathcal{Z}}(\cdot), \quad \mathbf{f_c} \approx \hat{\mathbf{f}}_{\mathbf{c}} = \mathrm{DeepRV}(\mathbf{z}, \mathbf{c}).$$

DeepRV is not restricted to any specific architecture—any network that accepts a latent variable and conditional variables, and outputs a matching stochastic process can serve as the decoder.

Algorithm 1 DeepRV GP workflow example

Fix the **spatial structure** of interest $\mathbf{x} = (x_1, \ldots, x_n)$, *e.g.* centroids of administrative units
Fix the **GP** of interest, *i.e.* a kernel $k_{\mathbf{c}}(\cdot, \cdot)$

Train DeepRV prior:
- Sample hyperparameters: $\mathbf{c} \sim p_{\mathcal{C}}(\cdot)$
- Sample GP realizations:
  - Generate $\mathbf{K_c} = k_{\mathbf{c}}(\mathbf{x}, \mathbf{x})$ over the spatial structure $\mathbf{x}$
  - Compute the Cholesky factor $\mathbf{L_c} = \mathrm{Cholesky}(\mathbf{K_c})$
  - Sample $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and set $\mathbf{f_c} = \mathbf{L_c z}$
- Forward pass: $\hat{\mathbf{f}}_{\mathbf{c}} = \mathrm{DeepRV}(\mathbf{z}, \mathbf{c})$
- Back propagate the loss: $\mathcal{L}_{\text{DeepRV}} = \mathrm{MSE}(\mathbf{f_c}, \hat{\mathbf{f}}_{\mathbf{c}})$

Perform Bayesian inference with MCMC of the overarching model, including latent variables and hyperparameters $\mathbf{c}$ in a drop-in manner using the trained network:

$$\mathbf{f_c} \sim \mathcal{GP}(\cdot, \cdot\, \mathbf{c}) \approx \hat{\mathbf{f}}_{\mathbf{c}} = \mathrm{DeepRV}(\mathbf{z}, \mathbf{c}), \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

3.2 SUPPORTED PRIORS

The training procedure relies relies on prior knowledge of a mapping function $T_{\mathbf{c}}$ that directly links the latent space to process realizations. While this design imposes more constraints than standard VAEs, it still enables DeepRV to encode complex priors. The most natural application of DeepRV

is GPs, a broad class of highly expressive and popular stochastic processes. To adapt a GP with a kernel $\mathbf{K_c}$ to DeepRV's training framework, we set the latent space $\mathcal{Z} = \mathbb{R}^n$, $\mathcal{D}_{\mathcal{Z}} = \mathcal{N}(\mathbf{0}, \mathbf{I})$, and define $T_{\mathbf{c}}(\mathbf{z}) = \mathbf{L_c z}$, where $\mathbf{L_c} = \text{Cholesky}(\mathbf{K_c})$. The distribution of $\mathbf{L_c z}$ when sampling $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is equivalent to $\mathcal{MVN}(\mathbf{0}, \mathbf{K_c})$, ensuring the correct mapping from the latent space to the GP realizations. A complete, generalized GP workflow example is provided in Algorithm 1.

Beyond GPs, DeepRV can also be applied to other spatial models, such as CAR, ICAR, and BYM, in a similar fashion. Eventually, we plan to expand this to any process that can be represented functionally as: $(\mathbf{z}, \mathbf{c}) \rightarrow \mathbf{f_c}$. This would enable modeling not just GPs, but whole hierarchical models or even black box simulators, although this is beyond the scope of the current study and left for future research. An example DeepRV workflow with a hierarchical model is provided in Appendix B.

4 Experiments

This section presents the experiments conducted on simulated datasets. These involve pretraining DeepRV priors across multiple GP kernels, evaluating the resulting models using Empirical Bayes, and performing inference. DeepRV's codebase is available in the following anonymous repository[1]. The API is designed to accept a GeoPandas (Jordahl et al., 2023)-compatible map, and the maps used in these experiments can be accessed via the map download link[2].

4.1 Architecture

In this paper, we implement DeepRV using a simple 2-layer MLP without dimensionality reduction. This choice maintains consistency with previous models for straightforward comparison and highlights that our primary contribution lies in the novel training process and decoder-only architecture. Despite its simplicity, our results show that even a basic DeepRV outperforms existing approaches and enables robust, accelerated inference.

4.2 Simulation study: UK LTLAs

4.2.1 Data

The geographical structure consists of 363 Lower Tier Local Authorities (LTLAs) from the UK, with data simulated using the centroids of each LTLA. We simulate data over this map because the number of locations ($n = 363$) is small enough to enable benchmarking against MCMC with GP sampling, while still representing a standard structure used in disease mapping. This allows us to evaluate whether our method is applicable to real-world data. The coordinates of the centroids are normalized to a [0,1] range and serve as spatial structure for DeepRV's GP workflow (see Algorithm 1). We also experimented with the CAR prior, which relies on an underlying graph structure of the data. An adjacency matrix is constructed by connecting neighboring LTLAs, ensuring that islands are linked to the nearest LTLA to maintain a fully connected graph.

4.2.2 Prior Pre-training

We trained the DeepRV and PriorCVAE models to emulate GP priors across four distinct kernels: RBF, Matérn-3/2, Matérn-1/2, Matérn-5/2, and the CAR model. For the RBF and Matérn kernels, we defined hyperparameter priors with constant variance and lengthscales uniformly distributed between 0.1 and 0.6 for RBF, and between 0.1 and 0.7 for the Matérn kernels. The lengthscale ranges for the RBF and Matérn kernels were chosen to produce comparable mean smoothness across the priors, with the RBF kernel being inherently smoother than the Matérn kernels. For the CAR kernel, we set τ to a constant value of 1 and α is sampled from $\text{Beta}(4, 1)$, where

$$\mathbf{f}_{\text{CAR},\tau,\alpha} \sim \mathcal{MVN}(\mathbf{0}, \mathbf{R}^{-}), \quad \mathbf{R} = \tau\left(\mathbf{D} - \alpha\mathbf{A}\right). \tag{1}$$

[1] https://anonymous.4open.science/r/DeepRV-F9CF/

[2] https://drive.google.com/file/d/1-3OCLlQgQxuM_B-pUid89eLsu9k8UWGY/view?usp=drive_link

Table 1: Test $\mathrm{MSE}(\mathbf{f}, \hat{\mathbf{f}})$

Model	**Parameters**	CAR	Matérn-1/2	Matérn-3/2	Matérn-5/2	RBF
PriorCVAE	794K	0.374 ± 0.001	0.177 ± 0.002	0.072 ± 0.004	0.056 ± 0.003	0.042 ± 0.003
DeepRV	265K	**0.023 ± 0.001**	**0.016 ± 0.000**	**0.011 ± 0.000**	**0.012 ± 0.000**	**0.020 ± 0.000**

Table 2: Empirical Bayes: MSE of ground truth vs. inferred conditional variable

Model	CAR (α - Eq 1)	Matérn-1/2 (length scale)	Matérn-3/2 (length scale)	Matérn-5/2 (length scale)	RBF (length scale)
PriorCVAE	**0.000 ± 0.000**	0.036 ± 0.015	0.025 ± 0.013	0.015 ± 0.002	0.005 ± 0.003
DeepRV	**0.000 ± 0.000**	**0.009 ± 0.007**	**0.000 ± 0.000**	**0.000 ± 0.000**	**0.000 ± 0.000**

Here, $\mathbf{D}$ represents the degree matrix of the adjacency matrix $\mathbf{A}$. We compared PriorCVAE and DeepRV by computing the MSE between the process realization $\mathbf{f}$, and the reconstructed realization $\hat{\mathbf{f}}$. We trained the models over 5 seeds with a batch size of 16, over 100,000 batches. Results of reconstruction MSE, and standard deviation are summarized in Table 1. Additionally, we provide input reconstruction examples in Appendix C.

To evaluate the pre-trained priors we employ an Empirical Bayes (EB) approach. Specifically, we generate samples from either the surrogate model or a reference GP model using predefined hyperparameters. These samples serve as pseudo-observations, and we subsequently optimize the hyperparameters to maximize the marginal likelihood (Williams & Rasmussen, 2006). The full procedure for sampling from DeepRV and estimating GP hyperparameters with kernel $\mathbf{K}$ is as follows:

$$\mathbf{c} \sim p_{\mathbf{c}}(\cdot), \quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \boldsymbol{\sigma} \sim \mathcal{N}(\mathbf{0}, \sigma\mathbf{I}), \quad \mathbf{f}_c \sim \mathrm{DeepRV}(\mathbf{c}, \mathbf{z}) + \boldsymbol{\sigma},$$

$$\hat{\mathbf{c}}, \hat{\sigma} = \mathrm{argmax}_{\hat{\mathbf{c}}, \hat{\sigma}} \left(-\frac{1}{2}\mathbf{f}_{\mathbf{c}}^{\top} \tilde{\mathbf{K}}_{\hat{\mathbf{c}}}^{-1} \mathbf{f}_{\mathbf{c}} - \frac{1}{2}\log|\tilde{\mathbf{K}}_{\hat{\mathbf{c}}}| - \frac{n}{2}\log 2\pi \right), \tilde{\mathbf{K}}_{\hat{\mathbf{c}}} = \mathbf{K}_{\hat{\mathbf{c}}} + \mathrm{Diag}(\hat{\boldsymbol{\sigma}})$$

This process allows us to assess how closely the priors, learned via the surrogate models, emulate the behavior of true GP priors. We generated 100 samples for each of 5 random seeds and computed the mean squared error (MSE) between the true conditional variable (lengthscale or α) and the estimated value obtained via EB. The results are presented in Table 2 and in Appendix D.

4.2.3 Inference

After training the priors, we assessed their ability to emulate a GP within a Numpyro model for inference. We simulated data across UK's LTLAs using the following Poisson model, which was also employed as our inference model:

$$\begin{aligned} \mathbf{c} &\sim p_{\mathcal{C}}(\cdot), \\ \mathbf{f}_{\mathbf{c}} &\sim \mathcal{GP}_{\mathbf{c}}(\cdot) \\ \beta &\sim \mathcal{N}(0, 1) \\ \boldsymbol{\lambda} &= \exp(\beta + \mathbf{f}_{\mathbf{c}}) \\ \mathbf{y}_{\mathbf{c}} &\sim \mathrm{Poisson}(\boldsymbol{\lambda}) \end{aligned} \tag{2}$$

During inference, the pre-trained prior was used as a substitute for Eq. 2, replacing the computationally intensive GP sampling during MCMC. We employed the No-U-Turn Sampler (NUTS) (Hoffman & Gelman, 2014) algorithm for MCMC implemented in Numpyro, running 4 chains with 4,000 warmup iterations and collecting 10,000 posterior samples per chain. We set the prior distributions of the length scale to be $\mathrm{Beta}(3, 7)$ for all distance-based GP kernels, and α prior to be $\mathrm{Beta}(4, 1)$ for CAR prior. The results are presented in Table 3, and Figure 2.

Table 3: Simulated data inference results from a single run. The inferred variables closest to the true values and the lowest inference times are **bolded**.

		CAR (α - Eq 1)	**Matérn-1/2** (length scale)	**Matérn-3/2** (length scale)	**Matérn-5/2** (length scale)	**RBF** (length scale)
	True Value	0.95	0.2	0.2	0.2	0.2
GP	Inferred Value	**0.807**	0.346	0.270	0.239	**0.196**
	ESS Inferred Variable	3,876	1,055	342	307	194
	$\hat{r}$ Inferred Variable	1.001	1.013	1.015	1.010	1.022
	MSE($\mathbf{y}$, $\bar{\mathbf{y}}_{\text{pred}}$)	0.280	0.297	0.526	0.601	0.581
	Mean ESS $\hat{\mathbf{f}}$	28,528	2,163	213	27	17
	Runtime (s)	889	4,284	35,526	37,299	34,750
PriorCVAE	ESS Inferred Variable	79,405	71,594	67,875	83,284	52,136
	$\hat{r}$ Inferred Variable	1.000	1.000	1.000	1.000	1.000
	MSE($\mathbf{y}$, $\bar{\mathbf{y}}_{\text{pred}}$)	0.375	0.334	0.542	0.625	0.595
	Mean ESS $\hat{\mathbf{f}}$	29,003	27,434	30,578	47,627	22,227
	Runtime (s)	236	387	515	559	430
DeepRV	Inferred Value	0.799	**0.267**	**0.241**	**0.208**	0.166
	ESS Inferred Variable	78,595	43,873	20,420	21,105	17,609
	$\hat{r}$ Inferred Variable	1.000	1.000	1.000	1.000	1.000
	MSE($\mathbf{y}$, $\bar{\mathbf{y}}_{\text{pred}}$)	0.280	0.298	0.529	0.603	0.588
	Mean ESS $\hat{\mathbf{f}}$	95,714	33,091	27,009	27,400	20,483
	Runtime (s)	**233**	**349**	**445**	**498**	**369**

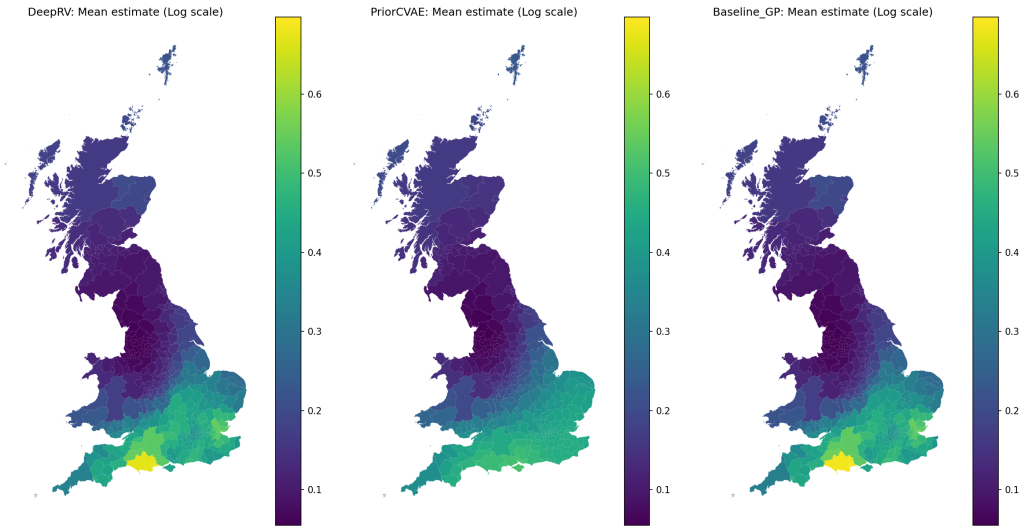


Figure 2: Simulated data posterior predictive $\boldsymbol{\lambda}$ means for the Matérn-1/2 kernel

5 Discussion and Future Work

The pre-training results summarized in Table 1, demonstrate that DeepRV surpasses PriorCVAE in reconstructing GP realizations. Table 2 further confirms that DeepRV's pre-trained prior more closely aligns with the emulated GP, a pattern that continues through inference (Table 3). We attribute this improvement to DeepRV's focused training objective—optimizing input reconstruction—rather than balancing reconstruction with encoder KL loss minimization. Moreover, inference on the simulated data (Table 3, Figure 2) shows that DeepRV produces posterior estimates closely matching GP sampling. In some cases, DeepRV even infers values closer to the true data-generating process, and achieves up to 30× speedups. This behavior is consistent across all tested kernels, as demonstrated in Appendix E.

We report the execution times using an NVIDIA RTX 5000 Ada Generation GPU (32GB RAM) to accelerate GP-based MCMC. Notably, DeepRV-based MCMC runs demonstrate significant speedups on CPUs, enabling parallelization for efficient local testing. However, full GP-based MCMC could not be benchmarked under the same conditions, as the sampling GPs proved too intensive to parallelize, preventing direct comparisons in the experiments section.

In the short term, we intend to implement DeepRV on 2023 UK cancer data, to validate its real-life applicability. Subsequently, we will characterize more precisely which priors DeepRV can encode to fully capture the potential of this framework. Further, we will explore DeepRv's ability to model full hierarchical structures, as demonstrated in Appendix 3, and evaluate its scalability to finer administrative divisions, such as UK Middle Layer Super Output Areas (MSOA) (Rashid et al., 2021). Long term, we plan to expand our library of pretrained priors to cover various administrative levels within the UK (e.g., LTLA, LAD, MSOA, postcode-level) and extend support to other regions.

This study faces some limitations. DeepRV improves efficiency but still relies on costly sampling methods to generate training data, particularly the cubic complexity of GP kernel computations, which can lead to out-of-memory (OOM) issues and hinder scalability when training with many locations. Like VAEs, it requires a predefined spatial structure, which can be restrictive in certain applications. However, unlike VAEs, DeepRV cannot encode arbitrary processes; instead, it relies on prior knowledge of a mapping function that directly links the latent space to process realizations. Attempts to benchmark against Laplace approximation and ADVI proved challenging due to their sensitivity to tuning, leading to unreliable inferences. Future work should explore these alternatives and Simulation-Based Inference for improved evaluation.

DeepRV represents a step toward more efficient and scalable probabilistic modeling by leveraging pre-trained priors without the need for full GP sampling. Its ability to accelerate inference while closely approximating the GPs rigor, and integrate seamlessly with PPLs highlights its potential for broader adoption. Future research will focus on refining its applicability across domains, improving its scalability, and benchmarking it against alternative inference strategies.

REFERENCES

Oriol Abril-Pla, Virgile Andreani, Colin Carroll, Larry Dong, Christopher J Fonnesbeck, Maxim Kochurov, Ravin Kumar, Junpeng Lao, Christian C Luhmann, Osvaldo A Martin, et al. Pymc: a modern, and comprehensive probabilistic programming framework in python. *PeerJ Computer Science*, 9:e1516, 2023.

Julian Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(2):192–225, 1974.

Julian Besag and Charles Kooperberg. On conditional and intrinsic autoregressions. *Biometrika*, 82 (4):733–746, 1995.

Julian Besag, Jeremy York, and Annie Mollié. Bayesian image restoration, with two applications in spatial statistics. *Annals of the institute of statistical mathematics*, 43:1–20, 1991.

Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul A. Szerlip, Paul Horsfall, and Noah D. Goodman. Pyro: Deep universal probabilistic programming. *J. Mach. Learn. Res.*, 20:28:1–28:6, 2019. URL `http://jmlr.org/papers/v20/18-403.html`.

Bob Carpenter, Andrew Gelman, Matthew D Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus A Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: A probabilistic programming language. *Journal of statistical software*, 76, 2017.

Perry de Valpine, Daniel Turek, Christopher Paciorek, Cliff Anderson-Bergman, Duncan Temple Lang, and Ras Bodik. Programming with models: writing statistical algorithms for general model structures with NIMBLE. *Journal of Computational and Graphical Statistics*, 26:403–413, 2017. doi: 10.1080/10618600.2016.1172487.

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.

K. Jordahl, J. Van den Bossche, M. Fleischmann, B. McBride, J. Wasserman, T. Toledo, M. Perry, C. Farmer, G.A. Hjelle, and J. Gerard. Geopandas: Python tools for geographic data. https://geopandas.org, 2023.

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

Finn Lindgren and Håvard Rue. Bayesian spatial modelling with r-inla. *Journal of statistical software*, 63(19), 2015.

James Lucas, George Tucker, Roger B Grosse, and Mohammad Norouzi. Don't blame the elbo! a linear vae perspective on posterior collapse. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/7e3315fe390974fcf25e44a9445bd821-Paper.pdf.

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.

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

Theo Rashid, James E Bennett, Christopher J Paciorek, Yvonne Doyle, Jonathan Pearson-Stuttard, Seth Flaxman, Daniela Fecht, Mireille B Toledano, Guangquan Li, Hima I Daby, et al. Life expectancy and risk of death in 6791 communities in england from 2002 to 2019: high-resolution spatiotemporal analysis of civil registration data. *The Lancet Public Health*, 6(11):e805–e816, 2021.

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, Swapnil Mishra, Samir Bhatt, Seth Flaxman, and H Juliette T Unwin. Deep learning and mcmc with aggvae for shifting administrative boundaries: mapping malaria prevalence in kenya. In *International Workshop on Epistemic Uncertainty in Artificial Intelligence*, pp. 13–27. Springer, 2023a.

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*, 2023b.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/8d55a249e6baa5c06772297520da2051-Paper.pdf.

Yuhta Takida, Wei-Hsiang Liao, Chieh-Hsin Lai, Toshimitsu Uesaka, Shusuke Takahashi, and Yuki Mitsufuji. Preventing oversmoothing in vae via generalized variance parameterization. *Neurocomputing*, 509:137–156, 2022. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2022.08.067. URL https://www.sciencedirect.com/science/article/pii/S0925231222010591.

Yixin Wang, David Blei, and John P Cunningham. Posterior collapse and latent variable nonidentifiability. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 5443–5455. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/2b6921f2c64dee16ba21ebf17f3c2c92-Paper.pdf.

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

A PRIORCVAE WORKFLOW

Algorithm 2 PriorCVAE (Semenova et al., 2023b) workflow

Fix the **spatial structure** of interest $\mathbf{x} = (x_1, \ldots, x_n)$, *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: $\mathbf{c} \sim p_{\mathcal{C}}(\cdot)$.
- Sample GP realizations: $\mathbf{f_c} \sim \mathcal{GP}_{\mathbf{c}}(\cdot)$, over the spatial structure $\mathbf{x}$
- Encode $\hat{\mathbf{z}}_\mu, \hat{\mathbf{z}}_\sigma = E_\gamma(\mathbf{f_c}, \mathbf{c})$, sample $\hat{\mathbf{z}} \sim \mathcal{N}(\hat{\mathbf{z}}_\mu, \hat{\mathbf{z}}_\sigma)$, and decode $\hat{\mathbf{f}}_\mathbf{c} = D_\psi(\hat{\mathbf{z}}, \mathbf{c})$.
- Back propagate the loss:$\mathcal{L}_{\text{CVAE}} = \frac{1}{\sigma^2_{\text{vae}}} \text{MSE}(\mathbf{f_c}, \hat{\mathbf{f}}_\mathbf{c}) + \text{KL}\left[\mathcal{N}(\hat{\mathbf{z}}_\mu, \hat{\mathbf{z}}_\sigma) || \mathcal{N}(\mathbf{0}, \mathbf{1})\right]$

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

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

B DEEPRV HIERARCHICAL MODEL TRAINING EXAMPLE

In this section, we describe how DeepRV can be used to emulate hierarchical model directly, for example consider the following Poisson model:

$$\begin{aligned}
\mathbf{c} &\sim p_{\mathcal{C}}(\cdot), \\
\mathbf{f_c} &\sim \mathcal{GP}_{\mathbf{c}}(\cdot) \\
\beta &\sim \text{Uniform}(-3, 3) \\
\boldsymbol{\lambda} &= \exp(\beta + \mathbf{f_c}) \\
\mathbf{y_c} &\sim \text{Poisson}(\boldsymbol{\lambda})
\end{aligned}$$

Where $\mathbf{f_c}$ is drawn from a Gaussian process with kernel $K_\mathbf{c}$. Algorithm 3, demonstrates how DeepRV can be used to directly encode the entire model, which can further accelarate inference.

Algorithm 3 Batch train step for an hierarchical Poisson model, based on a GP with kernel $k_\mathbf{c}(\cdot, \cdot)$

1: **Input:** locations $\mathbf{x}$, $p_{\mathcal{C}}(\cdot)$, N
2: $\{(\mathbf{c}_i, \mathbf{z}_i, \beta_i)\}_{i=1}^N \leftarrow$ sample N triplets from $(p_{\mathcal{C}}(\cdot), \mathcal{N}(\mathbf{0}, \mathbf{1_n}), \text{Uniform}(-\mathbf{3}, \mathbf{3}))$
3: $\mathbf{L}_i = \text{Cholesky}(k_{\mathbf{c_i}}(\mathbf{x}, \mathbf{x}))$
4: $\mathbf{f} \sim \text{Poisson}(exp(\mathbf{L}_i \mathbf{z}_i + \beta_i))$
5: $\hat{\mathbf{f}}_i \leftarrow \text{DeepRV}(\text{concat}(\mathbf{z}_i, \beta_i), \mathbf{c}_i)$
6: $\mathcal{L} \leftarrow \frac{1}{N} \sum_{i=1}^N \text{MSE}(\mathbf{f}_i, \hat{\mathbf{f}}_i)$
7: BackProp($\mathcal{L}$)

C INPUT RECONSTRUCTION FIGURES

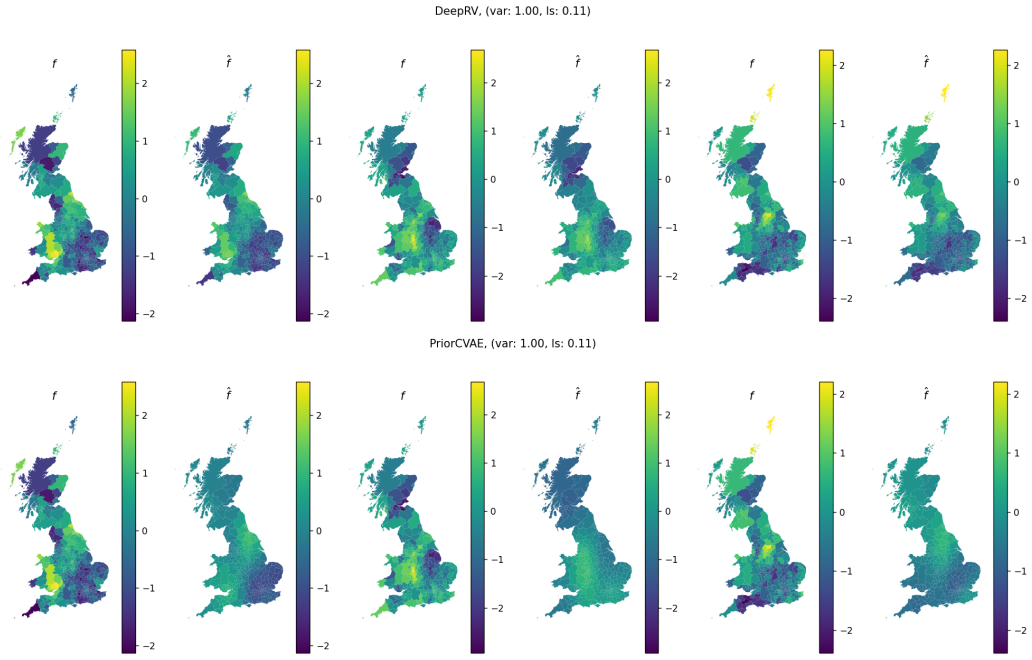


Figure 3: Input reconstruction examples Matérn 1/2 kernel GP. 3 examples ordered by the true data f and the reconstructed data $\hat{f}$.

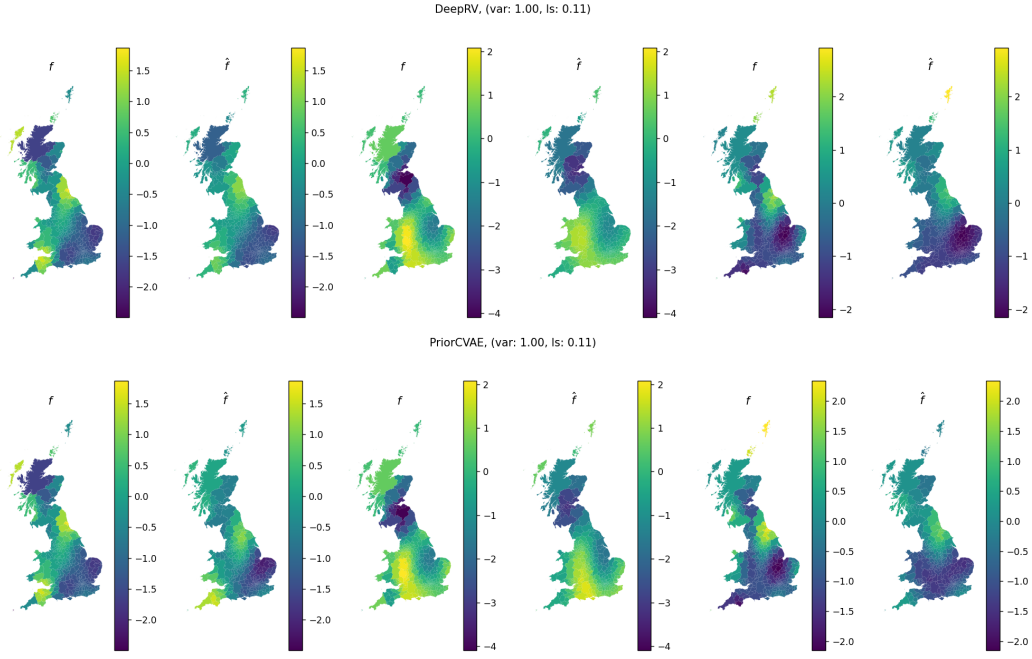


Figure 4: Input reconstruction examples RBF kernel GP. 3 examples ordered by the true data f and the reconstructed data $\hat{f}$.

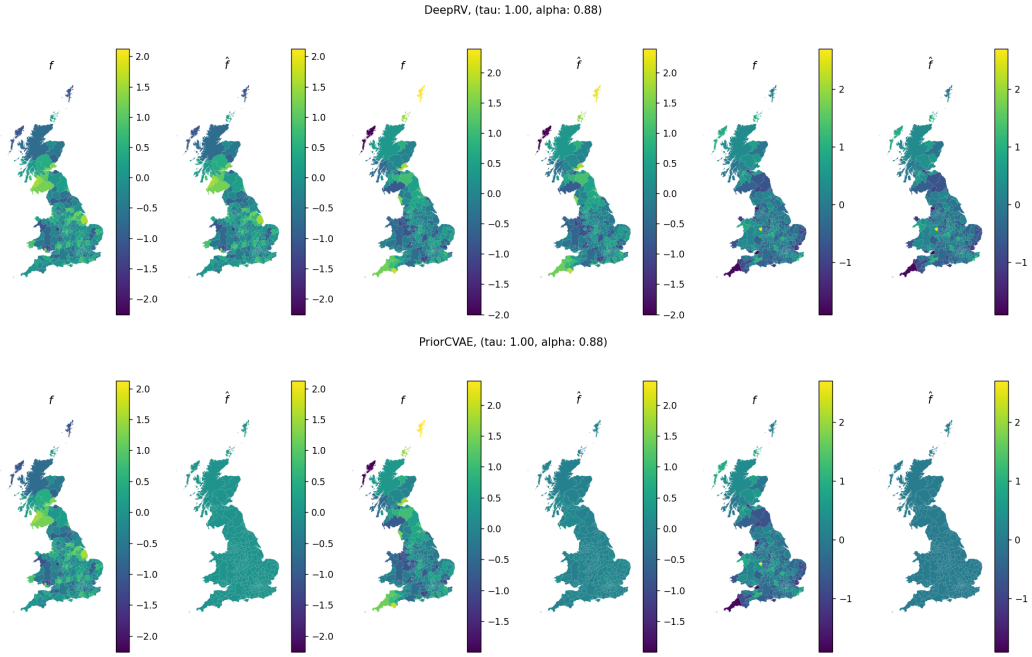


Figure 5: Input reconstruction examples CAR kernel GP. 3 examples ordered by the true data f and the reconstructed data $\hat{f}$.

D EMPIRICAL BAYES

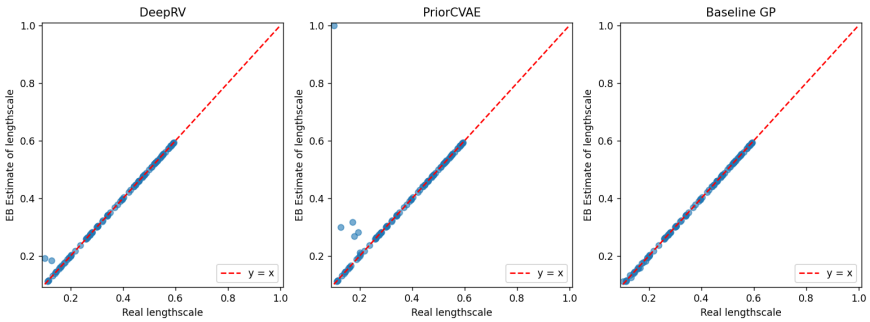


Figure 6: Empirical Bayes estimated vs. true length scales for RBF kernel GP.

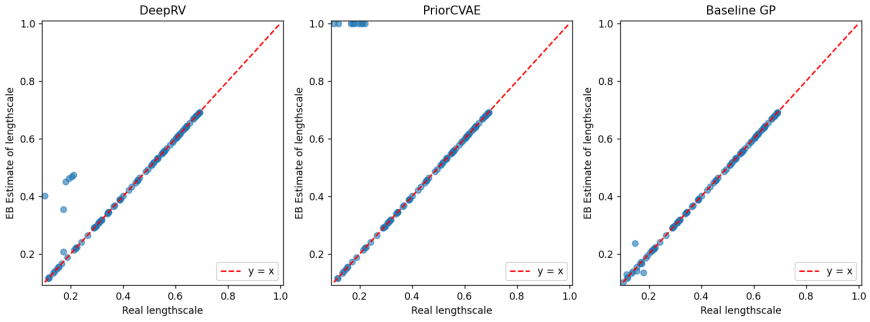


Figure 7: Empirical Bayes estimated vs. true length scales for Matérn 1/2 kernel GP.

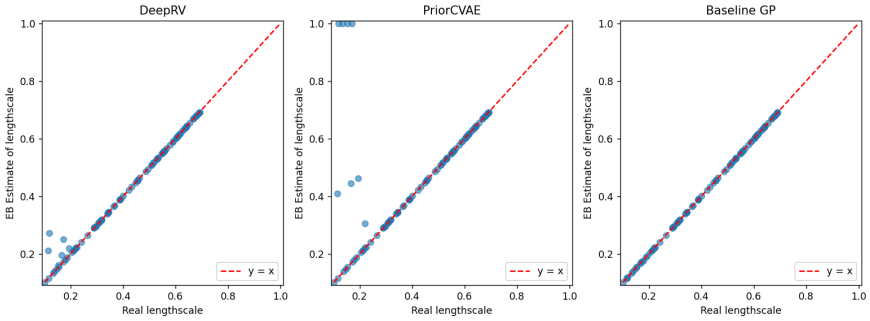


Figure 8: Empirical Bayes estimated vs. true length scales for Matérn 3/2 kernel GP.

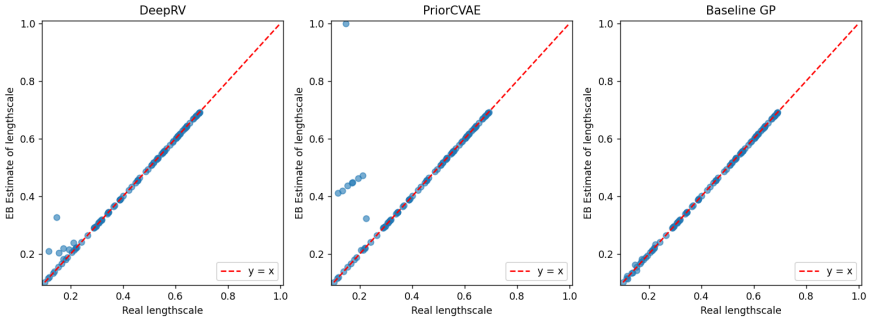


Figure 9: Empirical Bayes estimated vs. true length scales for Matérn 1/2 kernel GP.

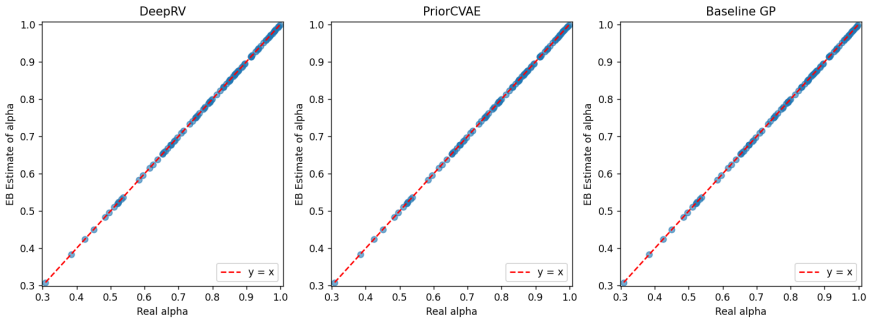


Figure 10: Empirical Bayes estimated vs. true length scales for CAR kernel.

E INFERENCE - SIMULATED DATA

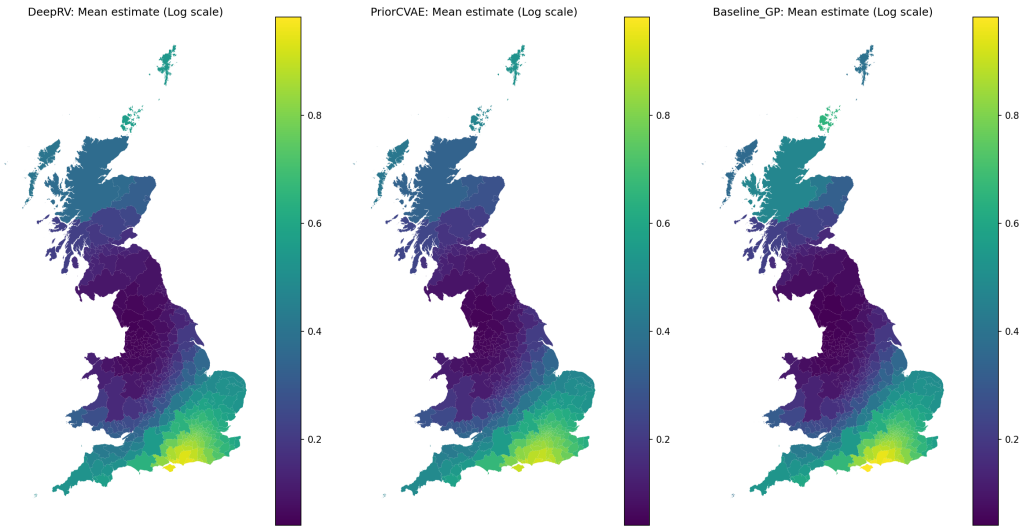


Figure 11: Simulated data posterior predictive $\boldsymbol{\lambda}$ means for the RBF kernel

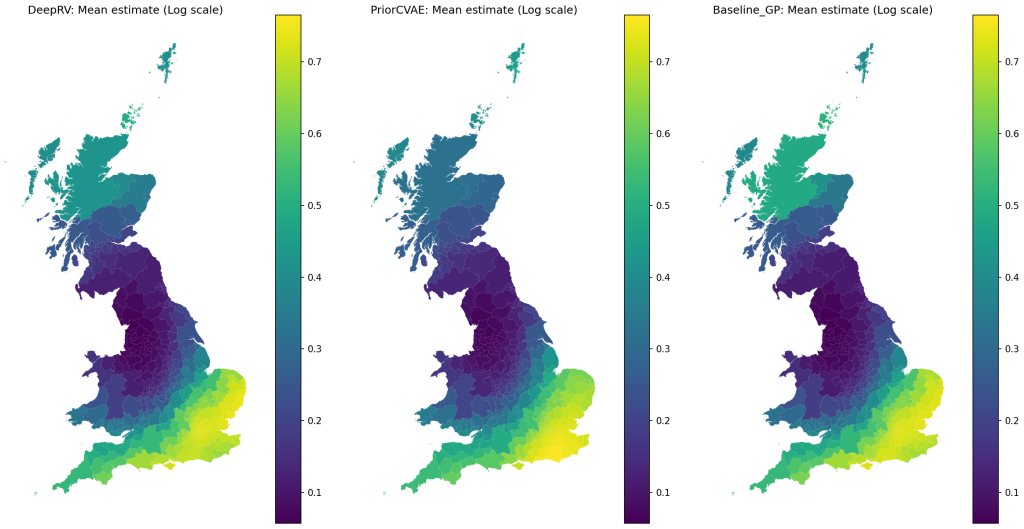


Figure 12: Simulated data posterior predictive $\boldsymbol{\lambda}$ means for the Maťern 3/2 kernel

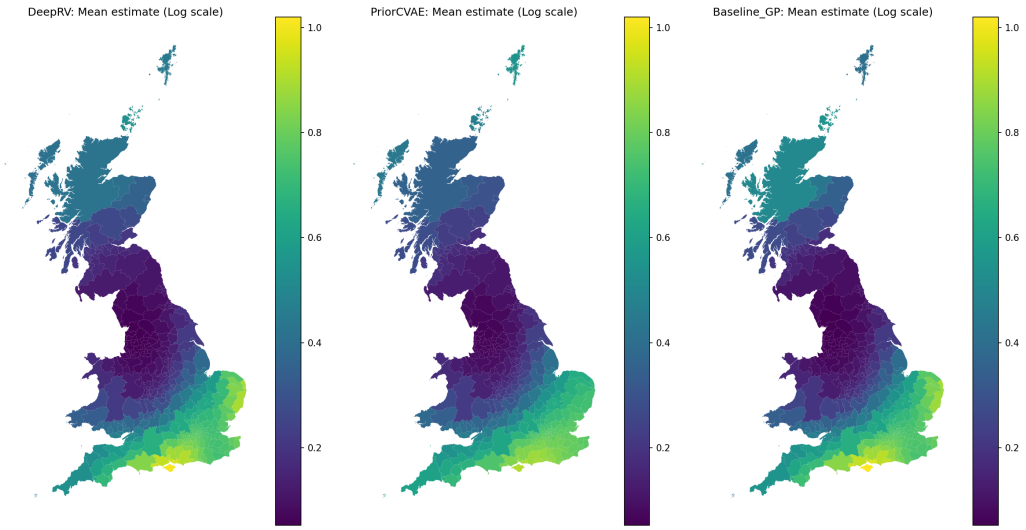


Figure 13: Simulated data posterior predictive $\boldsymbol{\lambda}$ means for the Mafern 5/2 kernel

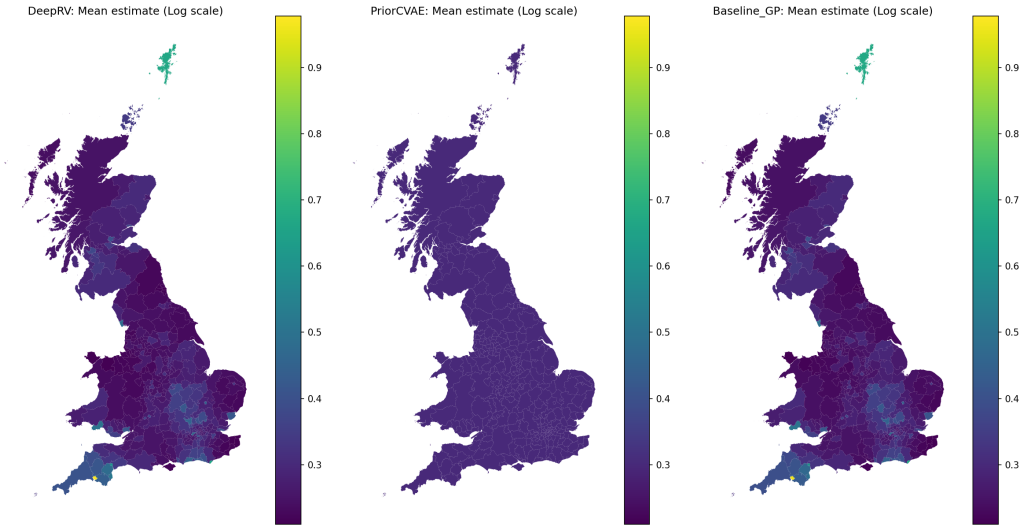


Figure 14: Simulated data posterior predictive $\boldsymbol{\lambda}$ means for the CAR model kernel