DeepKriging: Spatially Dependent Deep Neural Networks for Spatial Prediction

Wanfang Chen[1], Yuxiao Li[2], Brian J Reich[3] and Ying Sun[2]

[1] *East China Normal University*

[2] *King Abdullah University of Science and Technology*

[3] *North Carolina State University*

Supplementary Material

This supplementary material contains the proofs (Section S1), the settings for the DeepKriging network structure (Section S2), details of the simulation studies (Section S3), additional simulation studies (Section S4), distribution prediction and uncertainty quantification (Section S5), and the source codes and data for reproducible research (Section S6).

S1. Proofs

S1.1 Kriging prediction is linear in $\mathbf{x}_\phi(\mathbf{s}) = (\mathbf{x}(\mathbf{s}), \boldsymbol{\phi}(\mathbf{s}))^T$

Proof: Let the covariance matrix associated with the random process $\nu(\mathbf{s})$ be $\boldsymbol{\Sigma}^\nu$, where $\boldsymbol{\Sigma}^\nu_{i,j} = C(\mathbf{s}_i, \mathbf{s}_j)$. We can build a spatial random effect process $\mu(\mathbf{s}) = \boldsymbol{\phi}(\mathbf{s})^T\boldsymbol{\eta}$ with $\text{Cov}(\boldsymbol{\eta}) = \boldsymbol{\Phi}_R^{-1}\boldsymbol{\Sigma}^\nu(\boldsymbol{\Phi}_R^{-1})^T$, where $\boldsymbol{\Phi} = \{\boldsymbol{\phi}(\mathbf{s}_1), \ldots, \boldsymbol{\phi}(\mathbf{s}_N)\}$ is an $N \times K$ basis matrix with rank N, and $\boldsymbol{\Phi}_R^{-1}$ is the right inverse of $\boldsymbol{\Phi}$. From the result of Banerjee (1973), the right inverse exists. Hence, $\text{Cov}(\boldsymbol{\eta})$ is also a valid covariance matrix and $\text{Cov}(\boldsymbol{\mu}) = \text{Cov}(\boldsymbol{\nu}) = \boldsymbol{\Sigma}^\nu$, where $\boldsymbol{\mu} = \{\mu(\mathbf{s}_1), \ldots, \mu(\mathbf{s}_N)\}^T$ and $\boldsymbol{\nu} = \{\nu(\mathbf{s}_1), \ldots, \nu(\mathbf{s}_N)\}^T$.

Therefore, the Kriging prediction of $\mu(\mathbf{s})$ is the same as that of $\nu(\mathbf{s})$, i.e., $\widehat{Y}_{\mu}^{\mathrm{UK}}(\mathbf{s}_0) = \widehat{Y}_{\nu}^{\mathrm{UK}}(\mathbf{s}_0)$. On the other hand, based on the spatial random effect representation of $\mu(\mathbf{s})$, we can get the equivalent fixed rank Kriging prediction shown in the Equation (5) in the manuscript, such that $\widehat{Y}_{\mu}^{\mathrm{UK}}(\mathbf{s}_0) = \widehat{Y}_{\mu}^{\mathrm{FRK}}(\mathbf{s}_0) = \mathbf{x}(\mathbf{s}_0)^T\widehat{\boldsymbol{\beta}} + \boldsymbol{\phi}(\mathbf{s}_0)^T\widehat{\boldsymbol{\alpha}}$. $\square$

S1.2 Model capacity of DeepKriging compared to Kriging

Proof: Without loss of generality, we consider the mean squared loss in the prediction. The universal approximation theorem (Theorem 2.3.1 of Csáji (2001)) shows that $\forall f \in \mathbb{C}(\mathbf{x}_\phi)$, $\forall \varepsilon > 0$: $\exists n \in \mathbb{N}$ and certain choice of weights and biases, such that $|f(\mathbf{s}_0) - A_n f(\mathbf{s}_0)| < \varepsilon$, where $A_n f$ is the mapping from the standard multi-layer feed-forward networks with a single hidden layer that contains n hidden neurons. Then we have

$$\lim_{n\to\infty} |f(\mathbf{s}_0) - A_n^{\mathrm{opt}} f(\mathbf{s}_0)|^2 = 0, \tag{S1.1}$$

where $A_n^{\mathrm{opt}} f(\mathbf{s}_0)$ is the optimal neural network that approximating $f(\mathbf{s}_0)$.

In the context of spatial prediction, suppose the optimal prediction in $\mathbb{C}(\mathbf{x}_\phi)$ is $\widehat{Y}_{\mathbb{C}(\mathbf{x}_\phi)}^{\mathrm{opt}}(\mathbf{s}_0)$ that minimizes the mean squared loss, i.e., $\widehat{Y}_{\mathbb{C}(\mathbf{x}_\phi)}^{\mathrm{opt}}(\mathbf{s}_0) = \underset{\widehat{Y}(\mathbf{s}_0)\in\mathbb{C}(\mathbf{x}_\phi)}{\operatorname{argmin}} \mathbb{E}\{|\widehat{Y}(\mathbf{s}_0) - Y(\mathbf{s}_0)|^2\}$. Then, the mean squared loss of $\widehat{Y}_{\mathcal{F}_{\mathrm{DK}}}^{\mathrm{opt}}(\mathbf{s}_0)$ satisfies

$$0 \le \mathbb{E}\{|\widehat{Y}_{\mathcal{F}_{\mathrm{DK}}}^{\mathrm{opt}}(\mathbf{s}_0) - Y(\mathbf{s}_0)|^2\} - \mathbb{E}\{|\widehat{Y}_{\mathbb{C}(\mathbf{x}_\phi)}^{\mathrm{opt}}(\mathbf{s}_0) - Y(\mathbf{s}_0)|^2\} \le \mathbb{E}\{|\widehat{Y}_{\mathcal{F}_{\mathrm{DK}}}^{\mathrm{opt}}(\mathbf{s}_0) - \widehat{Y}_{\mathbb{C}(\mathbf{x}_\phi)}^{\mathrm{opt}}(\mathbf{s}_0)|^2\}.$$

Let $f(\mathbf{s}_0) = \widehat{Y}_{\mathbb{C}(\mathbf{x}_\phi)}^{\mathrm{opt}}(\mathbf{s}_0) \in \mathbb{C}(\mathbf{x}_\phi)$ in (S1.1), then $A_n^{\mathrm{opt}} f(\mathbf{s}_0) = \widehat{Y}_{\mathcal{F}_{\mathrm{DK}}}^{\mathrm{opt}}(\mathbf{s}_0)$ based

on the definition, hence $\lim_{n\to\infty} |\widehat{Y}^{\text{opt}}_{\mathcal{F}_{\text{DK}}}(\mathbf{s}_0) - \widehat{Y}^{\text{opt}}_{\mathbb{C}(\mathbf{x}_\phi)}(\mathbf{s}_0)|^2 = 0$. Also note that

$|\widehat{Y}^{\text{opt}}_{\mathcal{F}_{\text{DK}}}(\mathbf{s}_0) - \widehat{Y}^{\text{opt}}_{\mathbb{C}(\mathbf{x}_\phi)}(\mathbf{s}_0))|^2 \leq |\widehat{Y}^{\text{opt}}_{\mathbb{C}(\mathbf{x}_\phi)}(\mathbf{s}_0) - Y(\mathbf{s}_0)|^2 + |\widehat{Y}^{\text{opt}}_{\mathcal{F}_{\text{DK}}}(\mathbf{s}_0) - Y(\mathbf{s}_0)|^2 \leq 2|\widehat{Y}^{\text{opt}}_{\mathcal{F}_{\text{DK}}}(\mathbf{s}_0) - Y(\mathbf{s}_0)|^2$,

and $\mathbb{E}\{|\widehat{Y}^{\text{opt}}_{\mathcal{F}_{\text{DK}}}(\mathbf{s}_0) - Y(\mathbf{s}_0)|\}^2 < \infty$ by assumptions. Using the dominated convergence theorem, we have $\lim_{n\to\infty} \mathbb{E}\{|\widehat{Y}^{\text{opt}}_{\mathcal{F}_{\text{DK}}}(\mathbf{s}_0) - \widehat{Y}^{\text{opt}}_{\mathbb{C}(\mathbf{x}_\phi)}(\mathbf{s}_0)|^2\} = \mathbb{E}\{\lim_{n\to\infty} |\widehat{Y}^{\text{opt}}_{\mathcal{F}_{\text{DK}}}(\mathbf{s}_0) - \widehat{Y}^{\text{opt}}_{\mathbb{C}(\mathbf{x}_\phi)}(\mathbf{s}_0)|^2\} = 0$. So, $\widehat{Y}^{\text{opt}}_{\mathcal{F}_{\text{DK}}}(\mathbf{s}_0) = \widehat{Y}^{\text{opt}}_{\mathbb{C}(\mathbf{x}_\phi)}(\mathbf{s}_0) = \underset{Y(\mathbf{s})\in\mathbb{C}(\mathbf{x}_\phi)}{\text{argmin}}\ \mathbb{E}\{|\widehat{Y}(\mathbf{s}_0) - Y(\mathbf{s}_0)|^2\}\}$.

Since $\mathcal{F}_{\text{UK}} \subset \mathbb{C}(\mathbf{x}_\phi)$, we have

$$\mathbb{E}\{L(\widehat{Y}^{\text{opt}}_{\mathcal{F}_{\text{DK}}}(\mathbf{s}_0), Y(\mathbf{s}_0))\} = \mathbb{E}\{L(\widehat{Y}^{\text{opt}}_{\mathbb{C}(\mathbf{x}_\phi)}(\mathbf{s}_0), Y(\mathbf{s}_0))\} \leq \mathbb{E}\{L(\widehat{Y}^{\text{opt}}_{\mathcal{F}_{\text{UK}}}(\mathbf{s}_0), Y(\mathbf{s}_0))\}. \quad \square$$

S1.3 Proof of Equation (3.7)

Proof: Recall that the induced covariance function of $\hat{Y}_{\text{DK}}(\mathbf{s})$ is

$$C(\mathbf{s}, \mathbf{s}') = \mathbb{E}\{Y(\mathbf{s})Y(\mathbf{s}')\} = \sigma_b^2 + \sigma_w^2 \mathbb{E}\{a_j^1(\mathbf{s}) a_j^1(\mathbf{s}')\}.$$

According to Neal (1994),

$$\mathbb{E}\{a_j^1(\mathbf{s}) a_j^1(\mathbf{s}')\} = \frac{1}{2}[\mathbb{V}\text{ar}\{a_j^1(\mathbf{s})\} + \mathbb{V}\text{ar}\{a_j^1(\mathbf{s}')\}] - \frac{1}{2}\mathbb{E}\{[a_j^1(\mathbf{s}) - a_j^1(\mathbf{s}')]^2\}.$$

When $\mathbf{s}$ is close to $\mathbf{s}'$, we have $\psi_1(x) - \psi_1(y) = \alpha(x - y)$ for a smooth activation function, where α is a scaling coefficient. When the covariate vector $\mathbf{x}(\mathbf{s})$ is not available, we have $a_j^1(\mathbf{s}) = \psi_1(b_j^0 + \sum_{k=1}^K w_{jk}^0 \phi_k(\mathbf{s}))$. Thus we have

$$\mathbb{E}\{[a_j^1(\mathbf{s}) - a_j^1(\mathbf{s}')]^2\} = (\alpha\sigma_w)^2 \sum_{k=1}^K \{\phi_k(\mathbf{s}) - \phi_k(\mathbf{s}')\}^2.$$

Finally, the covariance function for nearby locations is

$$C(\mathbf{s}, \mathbf{s}') = \sigma_b^2 + \frac{\sigma_w^2}{2}[\mathbb{V}\text{ar}\{a_j^1(\mathbf{s})\} + \mathbb{V}\text{ar}\{a_j^1(\mathbf{s}')\}] - (\alpha\sigma_w^2)^2 \sum_{k=1}^K \{\phi_k(\mathbf{s}) - \phi_k(\mathbf{s}')\}^2$$

$$\equiv v(\mathbf{s}) + v(\mathbf{s}') - c\|\boldsymbol{\phi}(\mathbf{s}) - \boldsymbol{\phi}(\mathbf{s}')\|^2. \quad \square$$

S2. Settings for the DeepKriging network structure

The model settings for the DeepKriging network structure include the following considerations. First, when the sample size is small, we add a dropout layer to avoid overfitting. For a large dataset with Gaussian priors, dropout and other types of regularization are not very helpful. Second, batch-normalization is another useful strategy since the covariates typically have different units, and the scales of basis function may vary too much for irregularly spaced spatial data. Third, we normalize the covariates before we run the automated batch-normalization since the covariates and basis functions required different types of normalization. Last, when the data are irregularly spaced, the value of the compactly supported basis functions for some knots that are far apart from any other locations will be zero; in this case, we remove these knots and the relevant columns in the basis matrix.

To summarize, the default setting of DeepKriging network is as follows:

(1) Normalize (min-max normalization) the observed covariates $\mathbf{x}(\mathbf{s})$;

(2) Build the embedding layer using a 3 to 5 level multi-resolution radial basis functions $\boldsymbol{\phi}(\mathbf{s})$ with the corresponding basis matrix $\boldsymbol{\Phi}$; the form of the basis function and the choice of the number of basis functions K are illustrated at the beginning of Section 2.2;

(3) Remove the all-zero columns of the basis matrix $\boldsymbol{\Phi}$;

(4) Add the 1st dense layer with 100 hidden neurons and ReLu activation;

(5) Add the 1st dropout layer with 0.5 dropout rate;

(6) Add the 1st batch-normalization layer;

(7) Add the 2nd dense layer with 100 hidden neurons and ReLu activation;

(8) Add the 2nd dropout layer with 0.5 dropout rate;

(9) Add the 3rd dense layer with 100 hidden neurons and ReLu activation;

(10) Add the 2nd batch-normalization layer;

(11) Add the output layer.

The default number of epochs for DeepKriging is 200. For regression tasks, the loss function is set to be the mean squared error (MSE), while for classification tasks, the loss function is set to be the cross-entropy loss. For optimization, we use the Adam optimizer (Kingma and Ba, 2014). One can tune the hyper-parameters in the network (e.g., the value of K, the number of layers, the width of each layer, the dropout rate, the batch size and the number of epochs) in order for the optimization to converge and to achieve a better performance compared to other basic methods such as Kriging.

S3. Details of Simulation Studies

S3.1 DeepKriging on a 1-D Gaussian process

We first consider the performance of DeepKriging when data are simulated from a 1-D stationary GP, where the Kriging prediction is optimal. We also compare DeepKriging (with $x(s)$ and an embedding layer with basis functions of coordinate s as the input) to a DNN that does not account for the spatial dependence (with only $x(s)$ as the input), and a DNN that incorporates the spatial dependence by including directly the coordinates in the features (with $x(s)$ and coordinate s as the input). Specifically, the

data are generated from a GP with a constant mean: $z(s) = \mu + \nu(s) + \varepsilon(s)$, where $\mu = 1$, $\nu(s)$ is zero mean GP with an exponential covariance function $C(s, s') = \sigma^2 \exp\{-|s - s'|/\rho\}$, where the variance $\sigma^2 = 1$ and the range parameter $\rho = 0.1$, and $\varepsilon(s)$ is a Gaussian white noise with the nugget variance $\tau^2 = 0.01$. We generate 100 replicates for $\{z(s_1), \cdots, z(s_N)\}$ from the GP with $N = 1,000$ equally spaced locations over $[0, 1]$, with 800 locations randomly selected as training data and the remaining treated as testing data. In this example, there are no observed covariates except for the intercept, i.e., $x(s) = 1$ for any $s \in \mathbb{R}$. We also compare these models to Kriging prediction with the true exponential covariance function and that with an estimated Matérn covariance function where the smoothness parameter is set to 1.5. The above predictions related to Gaussian processes and deep learning are implemented using GPy (GPy, 2012) and Keras (Ketkar et al., 2017), respectively. Since we have Gaussian priors, dropout and other types of regularization for DeepKriging are not needed. The number of hidden layers is set to be 7 in DeepKriging, which yields the best performance (in terms of RMSE) for both the training and testing datasets among networks with 1 to 10 hidden layers (results not shown). Each hidden layer contains 100 neurons and uses the ReLu activation. The loss function is MSE, the number of epochs is set to be 100 and the batch size is set to be 32. A four-level multi-resolution model is used to generate the basis functions for DeepKriging, thus $K = 10 + 19 + 37 + 73 = 139$.

Figure S1 shows the prediction for one of the sample datasets using

each of the five prediction methods. Table S1 shows the root mean squared error (RMSE) and mean absolute percentage error (MAPE) on the both the training and testing sets over the 100 replicated samples.

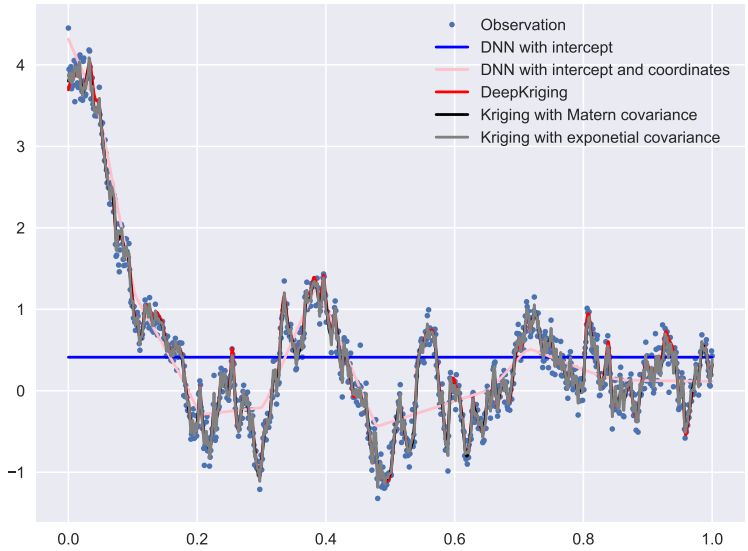

Figure S1: Prediction results for one simulated dataset generated from a GP. The blue dots are the simulated data (observations). The solid lines are the prediction curves from DNN with intercept only (blue line), DNN with intercept and coordinates (orange line), DeepKriging (red line), Kriging with the true exponential covariance function (grey line), and Kriging with an estimated Matérn covariance function (black line), respectively.

S3.2 DeepKriging on 2-D non-stationary data

In this section, we evaluate the performance of DeepKriging on 2-D non-stationary data so that the procedure is designed to resemble the real data application in Section 5 of the main manuscript. The goal is to predict values from the true process: $Y(\mathbf{s}) = \sin\{30(\bar{s} - 0.9)^4\}\cos\{2(\bar{s} - 0.9)\} + (\bar{s} - 0.9)/2$, where $\mathbf{s} = (s_x, s_y)^T \in \mathbb{R}^2$ and $\bar{s} = (s_x + s_y)/2$. A similar example is evaluated in many computer experiments (Ba et al., 2012), where

Table S1: Mean and standard deviations (SD) of the root mean squared errors (RMSEs) and mean absolute percentage errors (MAPEs) on both the training and testing sets across the 100 datasets from the five predictions of a Gaussian process. Kriging I and II are the Kriging prediction with the true exponential covariance function and that with an estimated Matérn covariance function, respectively. DeepKriging prediction is based on the MSE loss. DNN I and DNN II are the DNN prediction with intercept only and that with intercept and coordinates, respectively.

Models	Kriging I		Kriging II		DeepKriging		DNN I		DNN II	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Training set										
RMSE	**.068**	.002	.125	.006	.083	.009	.887	.183	.280	.029
MAPE	**.431**	1.639	.890	3.571	.851	3.834	5.158	9.538	1.779	4.475
Testing set										
RMSE	**.158**	.008	.164	.008	.258	.015	.885	.185	.292	.031
MAPE	**.536**	.504	.570	.548	.638	.540	3.123	2.755	.942	.843

both Kriging and neural networks are popularly applied. In our simulation, $N = 900$ observations are sampled on a $30\times$ 30 square grid of locations spanning $[0, 1]^2$; see Figure S2(a). This process presents obvious spatial non-stationarity; for example, the smoothness over the region $[0, 0.4]^2$ is significantly smaller than that over $[0.4, 1]^2$.

The network structures of DeepKriging as well as the baseline DNN are the default setting as stated in Section S2, except that the batch size is set to be 64, and the number of hidden layers is set to be 4 in DeepKriging, which yields the best performance (in terms of RMSE) for both the training and testing datasets among networks with 1 to 6 hidden layers (results not shown). A three-level multi-resolution model is used to generate the basis

functions for DeepKriging, thus $K = 10^2 + 19^2 + 37^2 = 1,830$. We use the 10-fold cross-validation method to show the performance of DeepKriging, Kriging with an estimated stationary covariance function and the baseline DNN with only coordinates s in the features. The boxplot of RMSEs are shown in Figure S2(b). The RMSEs and MAPEs on both the training and testing sets are shown in Table S2.

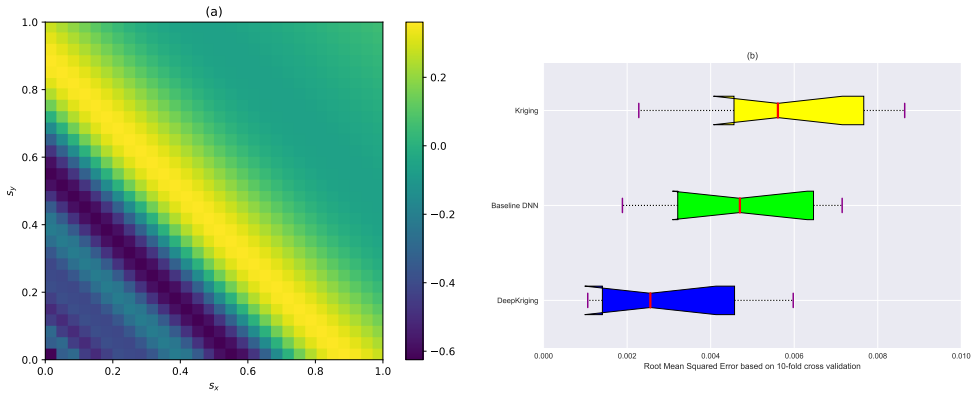

Figure S2: (a) Visualization of the simulated data generated from $Y(\mathbf{s}) = \sin\{30(\bar{s} - 0.9)^4\}\cos\{2(\bar{s} - 0.9)\} + (\bar{s} - 0.9)/2$, where $\mathbf{s} = (s_x, s_y)^T \in [0, 1]^2$ and $\bar{s} = (s_x + s_y)/2$. (b) Boxplots of the 10 RMSEs based on the 10-fold cross-validations from DeepKriging (blue), baseline DNN (green), and Kriging (yellow).

S4. Additional Simulations

S4.1 Choice of basis functions

As we have stated in the main manuscript, based on the KL theorem, the form of basis functions is not as important as the number of basis functions to approximate the spatial random effect $\nu(\mathbf{s})$. This can be supported by the additional simulations we conduct below.

Table S2: Model performance for both the training and testing sets based on 10-fold cross-validation. Data are generated from $Y(\mathbf{s}) = \sin\{30(\bar{s} - 0.9)^4\}\cos\{2(\bar{s} - 0.9)\} + (\bar{s} - 0.9)/2$, where $\mathbf{s} = (s_x, s_y)^T \in [0, 1]^2$ and $\bar{s} = (s_x + s_y)/2$. RMSEs and MAPEs are computed, and the mean and standard deviations (SD) for the 10 sets of validation errors are given. DeepKriging prediction is based on the MSE loss. The baseline DNN includes only coordinates $\mathbf{s}$ in the features. The Kriging prediction uses an estimated exponential covariance function.

Models	DeepKriging		Baseline DNN		Kriging	
	Mean	SD	Mean	SD	Mean	SD
Training set						
RMSE ($\times 10^{-3}$)	**.585**	.869	4.255	2.384	5.969e-4	7.287e-5
MAPE	**1.269**	1.313	5.296	3.170	5.749e-7	5.200e-8
Testing set						
RMSE ($\times 10^{-3}$)	**3.466**	3.417	6.934	7.672	8.552	9.562
MAPE	**5.330**	3.885	6.152	3.221	0.007	0.005

In Nychka et al. (2015), they stated that "the choice of the Wendland family of RBFs is not crucial and other compactly supported, positive definite functions will work." They used compactly supported kernels so that the key matrices are sparse to improve the computational efficiency. Our DeepKriging method does not need matrix operations, hence in principle we could use any positive definite kernels. For the 1-D simulation study of Section 4.1, here we replace our original choice of the Wendland kernel with a Gaussian kernel. Specifically, at a certain level of resolution, let $\{\mathbf{u}_j\}$, $j = 1, \ldots, m$, be a rectangular grid of points (i.e., the node points), and let θ be a scale parameter. The basis functions are then given by $\phi_j^*(\mathbf{s}) = \phi(\|\mathbf{s} - \mathbf{u}_j\|/\theta)$, where $\phi(d) = \exp\{-d^2\}$. We use the same ex-

perimental designs for the 1-D simulation as before. Figure S3 shows the prediction for one of the sample datasets using each of the six prediction methods. DeepKriging prediction with both basis function choices almost overlap with the optimal Kriging prediction. We then compare the RMSEs and MAPEs on both the training and testing datasets, as shown in Table S3 below. The results for predictions other than DeepKriging II (with the new basis functions) are only slightly different from our original results in Table S1, and the results are very close to each other between the two DeepKriging predictions. In addition, changing the basis functions do not change the conclusions overall; that is, for the testing set, the Kriging prediction with the true covariance function has the smallest RMSE as expected, and the performance of DeepKriging is comparable to the two Kriging predictions and outperforms the two naive DNN models. For the training set, both DeepKriging predictions are comparable to the optimal Kriging prediction, and they outperform the Kriging prediction with an estimated covariance function and the two naive DNN models in terms of both RMSE and MAPE.

Similarly for the 2-D simulation study in Section 4.2, replacing the radial basis functions using Wendland kernel with those using Gaussian kernel does not change the conclusion, if we compare Figure S4(b) with Figure S2(b), and compare Table S4 with Table S2. Specifically, for the testing set, in terms of RMSE, both DeepKriging predictions significantly outperforms Kriging in terms of RMSEs and MAPEs. In addition, the

baseline DNN is better than Kriging because the data are non-Gaussian and Kriging is no longer optimal. Moreover, the baseline DNN performs worse than DeepKriging as expected. The MAPE from DeepKriging is lower than the baseline DNN but higher than Kriging; this can happen since we are using MSE as the loss function in DeepKriging so it not necessarily possesses the lowest MAPE. For the training set, Kriging performs best in terms of both metrics since Kriging tends to underestimate such a variance, leading to a worse prediction on the testing dataset.

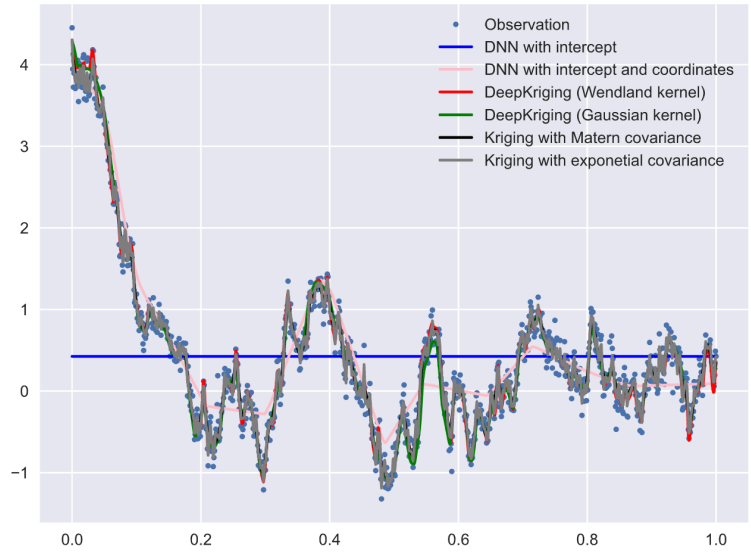

Figure S3: Prediction results for one simulated dataset generated from a GP. The blue dots are the simulated data (observations). The solid lines are the prediction curves from DNN with intercept only (blue line), DNN with intercept and coordinates (orange line), DeepKriging with Wenland kernel basis functions (red line), DeepKriging with Gaussian kernel basis functions (green line), Kriging with the true exponential covariance function (grey line), and Kriging with an estimated Matérn covariance function (black line), respectively.

Table S3: Mean and standard deviations (SD) of the root mean squared errors (RMSEs) and mean absolute percentage errors (MAPEs) on both the training and testing sets across the 100 datasets from the six predictions of a Gaussian process. Kriging I and II are the Kriging prediction with the true exponential covariance function and that with an estimated Matérn covariance function, respectively. DeepKriging I and II are the DeepKriging prediction with Wenland kernel and Gaussian kernel, respectively, based on the MSE loss. DNN I and DNN II are the DNN prediction with intercept only and that with intercept and coordinates, respectively.

Models	Kriging I		Kriging II		DeepKriging I		DeepKriging II		DNN I		DNN II	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Training set												
RMSE	**.063**	.002	.125	.006	.087	.011	.120	.008	.887	.183	.243	.025
MAPE	**.217**	.128	.456	.318	.274	.189	.450	.453	4.030	4.725	.915	.921
Testing set												
RMSE	**.160**	.008	.165	.008	.197	.018	.187	.015	.884	.186	.254	.029
MAPE	**.573**	.541	.603	.583	.676	.639	.676	.635	3.495	2.747	.880	.781

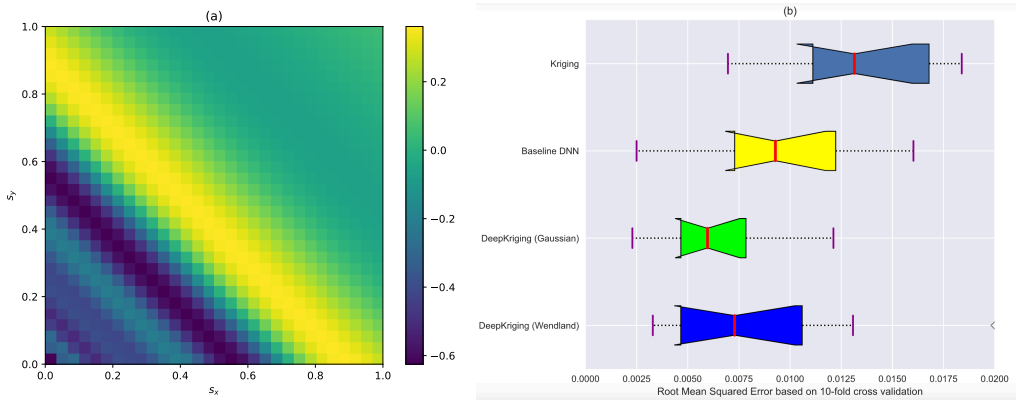

Figure S4: (a) Visualization of the simulated data generated from $Y(\mathbf{s}) = \sin\{30(\bar{s} - 0.9)^4\}\cos\{2(\bar{s} - 0.9)\} + (\bar{s} - 0.9)/2$, where $\mathbf{s} = (s_x, s_y)^T \in [0, 1]^2$ and $\bar{s} = (s_x + s_y)/2$. (b) Boxplots of the 10 RMSEs based on the 10-fold cross-validations from DeepKriging with Wendland kernel (blue), DeepKriging with Gaussian kernel (green), baseline DNN (yellow), and Kriging (gray).

Table S4: Model performance for both the training and testing sets based on 10-fold cross-validation. Data are generated from $Y(\mathbf{s}) = \sin\{30(\bar{s}-0.9)^4\}\cos\{2(\bar{s}-0.9)\} + (\bar{s}-0.9)/2$, where $\mathbf{s} = (s_x, s_y)^T \in [0,1]^2$ and $\bar{s} = (s_x + s_y)/2$. RMSEs and MAPEs are computed, and the mean and standard deviations (SD) for the 10 sets of validation errors are given. DeepKriging I and II are the DeepKriging predictions with Wenland kernel and Gaussian kernel, respectively, based on the MSE loss. The baseline DNN includes only coordinates $\mathbf{s}$ in the features. The Kriging prediction uses an estimated exponential covariance function.

Models	DeepKriging I		DeepKriging II		Baseline DNN		Kriging	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Training set								
RMSE ($\times 10^{-3}$)	**.526**	.062	**1.382**	.082	6.946	3.946	1.757e-4	2.399e-4
MAPE	**.732**	.838	**1.386**	.622	9.634	6.389	5.152e-7	6.624e-8
Testing set								
RMSE ($\times 10^{-3}$)	**8.431**	4.927	**6.578**	3.065	9.987	5.375	15.135	7.804
MAPE	**10.520**	5.397	**5.798**	2.361	10.875	7.494	.098	.088

S4.2 DeepKriging is non-linear

It is important to investigate how the DeepKriging prediction is linked with the observations. We generate 100 observations generated by $z(s) = Y(s)\mathbb{1}_{\{Y(s)>0\}} + \varepsilon(s)$, where $Y(s) = 10\cos(20s)$, the coordinates s are regularly located in $[0,1]$, and $\varepsilon(s) \overset{iid}{\sim} N(0,1)$. The simulated data $z(s)$ and signal $Y(s)$ are shown in Figure S5(a). The figure also shows the prediction results from both Kriging and DeepKriging, which are almost identical.

To examine the non-linear relationship between a training observation and the prediction, we replace the observation $z(s)$ at $s = 50$ by $m = 50$ different values and drop the observation $z(s+1)$ in the model training, and obtain the prediction $\hat{Y}(s+1)$ by Kriging or DeepKriging. By doing

so, we are able to test the relationship and sensitivity of $z(50)$ on $\hat{Y}(51)$ for both methods. The results in Figure S5(b) show that the Kriging prediction is linear in observations, while the DeepKriging provides an obviously nonlinear prediction in observations. This simulation study shows that although the Kriging and DeepKriging predictions are almost identical, the underlying relationships between the prediction and observations are totally different.

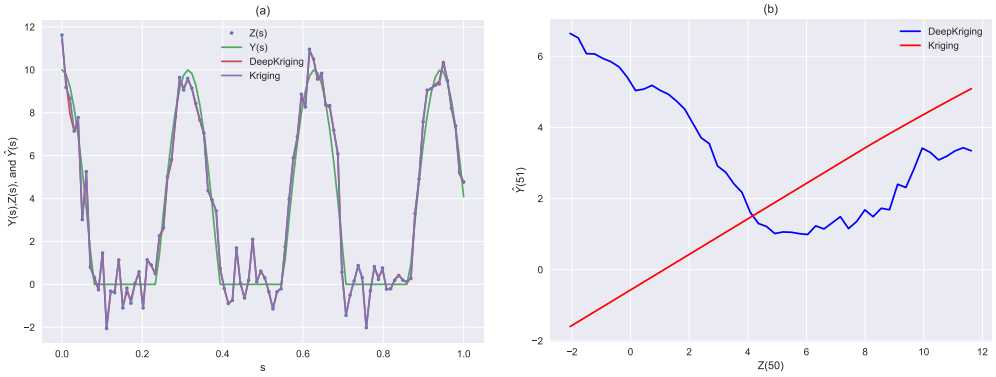

Figure S5: (a) Visualization of the simulated data and predictions. The observations (blue dots) z(s) are generated by the the signal (green line) $Y(s)\mathbb{1}_{\{Y(s)>0\}} + \varepsilon(s)$, where $Y(s) = 10\cos(20s)$, with a standard Gaussian noise. The predictions of Kriging (purple line) and DeepKriging (red line) are almost identical. (b) The relationship and sensitivity of $z(50)$ on $\hat{Y}(51)$ from Kriging (red line) and DeepKriging (blue line) prediction.s The x-axis shows 50 different values of $z(50)$ in the model training and the y-axis shows 50 predicted values of $Y(51)$ provided that $z(51)$ is not used.

S4.3 Computational time of DeepKriging

Based on the same simulation setting in Section S4.2, we investigate the computational time of DeepKriging compared to Kriging with different sample sizes N. Figure S6 shows the results. Although Kriging is faster for

small sample sizes ($N < 1,500$), DeepKriging is much more scalable when the sample size increases. For example, when $N = 12,800$, which is the largest sample size we have considered, it takes more than 1.5 hours (5,663 seconds) to train a Kriging model, but DeepKriging only costs 3.5 minutes (214 seconds) without GPU acceleration and 1.5 (94 seconds) minutes with a Tesla P100 GPU. Note that the GPU we use is freely accessed in Kaggle. For a more powerful GPU, the computational cost will be further reduced.

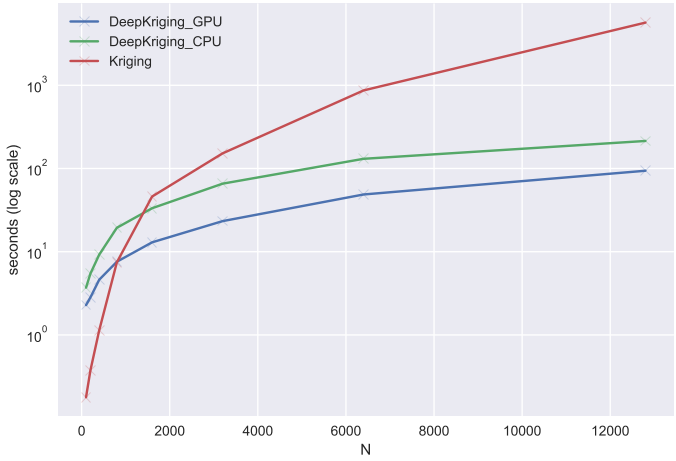

Figure S6: Computational time in seconds ($\log_{10}$ scale) of DeepKriging and Kriging for different sample sizes N. The green line is the computational time of Kriging. The blue and green lines are the computational time of DeepKriging based on a Tesla P100 GPU and a 2.5 GHz Intel Core i7 CPU, respectively.

S5. Distribution prediction and uncertainty quantification of DeepKriging

An ideal spatial prediction not only provides point prediction, but also distributional information such as quantiles or the density function to quantify

uncertainties, risks and extreme values (Diggle et al., 2007). Traditional DNNs cannot provide the uncertainty information at the predicted spatial locations. Recently, several methods have been proposed to overcome this problem by predicting the entire probabilistic distribution.

Gal and Ghahramani (2016) and Posch et al. (2019) applied Bayesian inference methodologies to neural networks to predict uncertainties via the posterior distribution. However, these methods cannot be applied directly to spatial data. Some studies try to combine deep learning and GPs which can be potentially applied in spatial prediction (Damianou and Lawrence, 2013; Lee et al., 2017; Kumar et al., 2018; Zammit-Mangion et al., 2019; Blomqvist et al., 2019). These methods are often called deep Gaussian processes that characterize deep learning in the framework of GPs via Bayesian learning (Neal, 1996). Based on the correspondence between an infinitely wide DeepKriging and GPs as illustrated in Section 3.3 in the main manuscript, we can potentially compute the covariance function for the induced GPs and then use the resulting GPs to perform Bayesian inference for wide DeepKriging networks. However, it is infeasible to train an infinitely wide DNN in practice, and Lee et al. (2017) found that the test performance increases as finite-width trained networks are made wider and more similar to a GP, and thus that GP predictions typically outperform those of finite-width networks. The related computation burden is another issue for applications in large datasets.

The Bayesian methods mentioned above require configurations on the

prior distributions for hyper-parameters. Li et al. (2019) proposed a completely distribution-free method for density estimation called deep distribution regression (DDR), where the density was approximated using histograms with discrete bins, and the discretized density was estimated by multi-class classification using neural networks. Although the method was designed for a regression problem, we can extend the idea to our DeepKriging framework for predictive distribution estimation. The idea is straightforward and the method is easy to implement. We call the method deep distribution spatial prediction (DDSP), as illustrated below.

S5.1 Deep Distribution Spatial Regression

We quantify the uncertainty of DeepKriging prediction by the conditional distribution of $Y(\mathbf{s}_0)|\mathbf{z}, \mathbf{x}_\phi(\mathbf{s}_0)$ at an unobserved location $\mathbf{s}_0$. We denote the probability density function (PDF) as $f_{\text{UQ}}(y|x)$, whose support is $[l, u]$, with l and u being the lower and upper bound, respectively. The interval can be partitioned into $M+1$ bins, $T_m = [c_{m-1}, c_m)$, by M cut-points such that $c_0 < c_1 < \cdots < c_M < c_{M+1}$, where $c_0 = l$ and $c_{M+1} = u$. Let $|T_m|$ be the length of the m-th bin and $p_m(\mathbf{s}_0) = \mathbb{P}\{Y(\mathbf{s}_0) \in T_m|\mathbf{z}, \mathbf{x}_\phi(\mathbf{s}_0)\}$ be the conditional probability that $Y(\mathbf{s}_0)$ falls into the m-th bin. Then $f_{\text{UQ}}(y|x)$ is approximated by $M+1$ constant functions, $p_m(\mathbf{s}_0)/|T_m|, m = 1, \ldots, M+1$. The density prediction is specified as:

$$\hat{f}_{\text{UQ}}(y|x) = \sum_{m=1}^{M+1} \frac{\widehat{p}_m(\mathbf{s}_0)}{|T_m|} \mathbb{1}\{y \in T_m\}. \tag{S5.2}$$

The crucial step in Equation (S5.2) is to estimate the bin probabilities

$\{\widehat{p}_1(\mathbf{s}_0), \ldots, \widehat{p}_{M+1}(\mathbf{s}_0)\}$ using a classification model, so that neural networks can be applied. Li et al. (2019) suggested different ways for loss functions and bin partitioning. The most natural way is to use multi-class classification with fixed bins. In the output layer, a softmax function is applied to ensure that $\{\widehat{p}_1(\mathbf{s}_0), \ldots, \widehat{p}_{M+1}(\mathbf{s}_0)\}$ constitutes a valid probability vector. The loss function for this case is the negative multinomial log-likelihood function, which is equivalent to the multi-class cross entropy loss: $\sum_{n=1}^{N} \sum_{m=1}^{M+1} \mathbb{1}\{z(\mathbf{s}_n) \in T_m\} \log\{p_m(\mathbf{s}_n)\}$.

In practice, however, the estimation of a continuous density function by the multi-class fixed classification is sensitive to the choice of bins. Therefore, for DDSP in DeepKriging, we use the improved option with joint binary cross entropy loss function (JBCE) and ensembles. The JBCE function is specified as

$$\mathrm{JBCE} = \sum_{\mathrm{n}=1}^{\mathrm{N}} \sum_{\mathrm{m}=1}^{\mathrm{M}} \left[\mathbb{1}\{\mathrm{z}(\mathbf{s}_\mathrm{n}) \leq \mathrm{c_m}\} \log\{\mathrm{F}(\mathrm{c_m}; \mathbf{s}_\mathrm{n})\} + \mathbb{1}\{\mathrm{z}(\mathbf{s}_\mathrm{n}) > \mathrm{c_m}\} \log\{1 - \mathrm{F}(\mathrm{c_m}; \mathbf{s}_\mathrm{n})\}\right], \tag{S5.3}$$

where $F(c_m; \mathbf{s}_n) = \sum_{i=1}^{m} p_i(\mathbf{s}_n)$ and $c_m, m = 1, \ldots, M$ are the M cut-points.

The DDSP may provide a non-smooth density. Thus, an ensemble method can be applied for further adjustment by fitting I independent classifications and computing the average of classifiers. Algorithm 1 provides the procedure of obtaining density prediction of DeepKriging with the ensemble of random partitioning:

Note that although the density regression provides an uncertainty quantification without any distribution assumption, it may bring new uncertain-

Algorithm 1: The algorithm of density prediction of DeepKriging with ensemble random partitioning. We ensemble I independent classifications, for each of which M bins are chosen randomly.

for i *=1:I* **do**

Draw M cut-points from Uniform(0,1);

Sort the cut-points as $c_{i1}, \ldots, c_{iM}$;

Assign $Y(\mathbf{s}_0)$ to one of the M bins, where the $m-th$ bin is

$T_{im} = [c_{i,m-1}, c_{im})$;

Train the classifier to estimate the probabilities $\widehat{p}_{i1}(\mathbf{s}_0), \ldots, \widehat{p}_{i,M+1}(\mathbf{s}_0)$;

end

Result: $\hat{f}_{\mathrm{UQ}}(y|x) = \sum_{i=1}^{I} \sum_{m=1}^{M+1} \frac{\widehat{p}_{im}(\mathbf{s}_0)}{|T_{im}|} \mathbb{1}\{y \in T_{im}\}$

ties from the neural networks. Therefore, the GP process representation may be a better way for uncertainties if it is quantified by the standard deviation, even though the computational burden will increase.

In addition, the density prediction requires further computations if we apply ensembles, where each step in the ensemble will implement classification separately with a new choice of random cut-points. However, the main objective of the ensemble in the density prediction is to obtain a smooth density. If the research goal is to get uncertainties using quantiles without the density curve, then the ensemble is not always needed. Another hyperparameter we need to specify in the density prediction is the bin size $M + 1$. Typically, large M will provide a more complex pattern of the density, and small M will give a more stable estimation. In practice, we can follow the Freedman–Diaconis rule (Freedman and Diaconis, 1981) as a robust estimation in the histogram approximation such

that $M = \left\lfloor \{\max(\mathbf{z}) - \min(\mathbf{z})\} \frac{\sqrt[3]{N}}{2 \times \mathrm{IQR}(\mathbf{z})} \right\rfloor$, where $\mathrm{IQR}(\mathbf{z})$ is the interquartile range of the observations.

S5.2 Uncertainty Quantification: A Simulation Study

The simulated data is generated by a 1-D Gaussian mixture model:

$$z(s) = \{\sin(5s) + 0.7 + \tau_1(s)\}\pi(s) + \{2\sin(8s) + \tau_2(s))\}(1 - \pi(s)),$$

where $\tau_1(s) \overset{iid}{\sim} N(0, 0.2^2)$, $\tau_2(s) \overset{iid}{\sim} N(0, 0.3^2)$, $\pi(s) \overset{iid}{\sim}$ Bernoulli(0.5), and $\tau_2(s)$, $\tau_2(s)$, and $\pi(s)$ are mutually independent. Observations are drawn from $2,500$ regularly spaced locations in $[0, 1]$, out of which $2,000$ are training samples and 500 are testing samples.

Figure S7 shows the prediction results on both training and testing samples. The advantage of DeepKriging is obvious for the uncertainty quantification. The process is heteroscedastic with an obviously large variance when s is larger than 0.8. The heteroscedasticity is well captured by the DeepKriging but missed by the Kriging. Moreover, the prediction band of the Kriging is too narrow for large s and two large for small s.

Numerical evaluations on testing samples are further used for comparison based on the root mean squared error (RMSE) and average quantile loss (AQTL). AQTL is defined by $\mathrm{AQTL} = \sum_{\mathrm{t}=1}^{99} \sum_{\mathrm{n}=1}^{\mathrm{N}} (\{\mathrm{z}(\mathrm{s_n}) - \widehat{\mathrm{Q}}(\mathrm{t}/100)\}[\mathrm{t}/100 - \mathbb{1}\{\mathrm{z}(\mathrm{s_n}) \leq \widehat{\mathrm{Q}}(\mathrm{t}/100)\}])$, where $\widehat{Q}(t/100)$ is the estimated t-th percentile. The results show that the RMSE of the Kriging (0.490) is comparable to DeepKriging (0.485). But, in terms of the quantile loss, the AQTL of the Kriging (60.47) is much larger than that of DeepKriging (0.12), which fur-

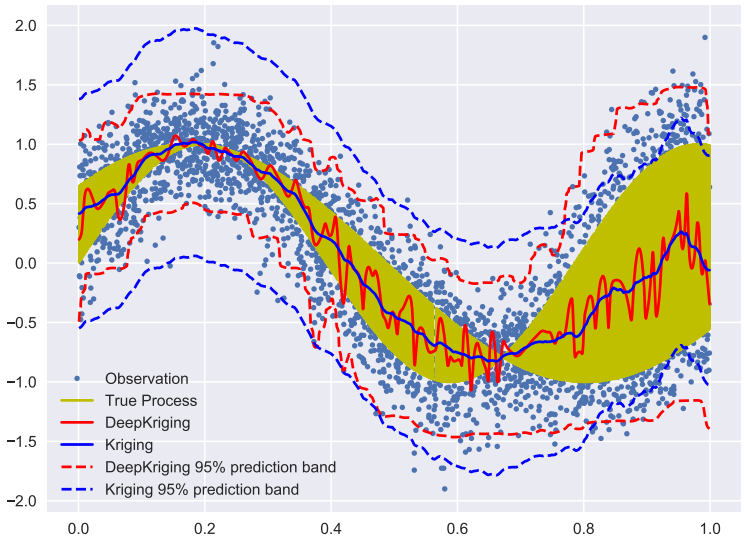

Figure S7: Prediction results for a Gaussian mixture model. The blue dots are the observations simulated from the Gaussian mixture model. The solid lines are the true process (yellow) and the predictions from DeepKriging (red) and Kriging (blue), respectively. The dashed lines represent the 95% prediction bands from DeepKriging (red) and Kriging (blue), respectively.

ther implies that Kriging is only a good estimation in terms of the mean squared error which reflects the mean and variance, but cannot reflect other important properties such as quantiles.

S6. Source codes and data

- The source codes can be found in the following Github repository: https://github.com/aleksada/DeepKriging.

- The methods are implemented in Keras, a library of both Python and R. Most of the codes for this work are written in Python. We also

provide some simple examples of DeepKriging using Keras in R.

- The PM2.5 station data from AQS used in the application can be found in https://aqs.epa.gov/aqsweb/airdata/download_files.html.
- The meteorological data from NARR used in the application can be found in https://psl.noaa.gov/data/gridded/data.narr.html.

References

Ba, S., Joseph, V. R., et al. (2012). Composite gaussian process models for emulating expensive functions. *The Annals of Applied Statistics*, 6(4):1838–1860.

Banerjee, K. S. (1973). Generalized inverse of matrices and its applications. *Technometrics*, 15(1):197–197.

Blomqvist, K., Kaski, S., and Heinonen, M. (2019). Deep convolutional gaussian processes. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 582–597. Springer.

Csáji, B. C. (2001). Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, 24:48.

Damianou, A. and Lawrence, N. (2013). Deep gaussian processes. In *Artificial Intelligence and Statistics*, pages 207–215.

Diggle, P. J., Thomson, M. C., Christensen, O., Rowlingson, B., Obsomer, V., Gardon, J., Wanji, S., Takougang, I., Enyong, P., Kamgno, J., et al.

(2007). Spatial modelling and the prediction of loa loa risk: decision making under uncertainty. *Annals of Tropical Medicine & Parasitology*, 101(6):499–509.

Freedman, D. and Diaconis, P. (1981). On the histogram as a density estimator: L2 theory. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 57(4):453–476.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059.

GPy (since 2012). GPy: A Gaussian process framework in python. `http://github.com/SheffieldML/GPy`.

Ketkar, N. et al. (2017). *Deep Learning with Python*. Springer.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kumar, V., Singh, V., Srijith, P., and Damianou, A. (2018). Deep gaussian processes with convolutional kernels. *arXiv preprint arXiv:1806.01655*.

Lee, J., Bahri, Y., Novak, R., Schoenholz, S. S., Pennington, J., and Sohl-Dickstein, J. (2017). Deep neural networks as gaussian processes. *arXiv preprint arXiv:1711.00165*.

Li, R., Bondell, H. D., and Reich, B. J. (2019). Deep distribution regression. *arXiv preprint arXiv:1903.06023*.

REFERENCES

Neal, R. M. (1994). Priors for infinite networks. *Technical Report*.

Neal, R. M. (1996). *Bayesian Learning for Neural Networks*, volume 118. Springer Science & Business Media.

Nychka, D., Bandyopadhyay, S., Hammerling, D., Lindgren, F., and Sain, S. (2015). A multi-resolution Gaussian process model for the analysis of large spatial datasets. *Journal of Computational and Graphical Statistics,* 24(2):579–599.

Posch, K., Steinbrener, J., and Pilz, J. (2019). Variational inference to measure model uncertainty in deep neural networks. *arXiv preprint arXiv:1902.10189*.

Zammit-Mangion, A., Ng, T. L. J., Vu, Q., and Filippone, M. (2019). Deep compositional spatial models. *arXiv preprint arXiv:1906.02840*.