Math Geosci
https://doi.org/10.1007/s11004-021-09988-0

SPECIAL ISSUE

Bayesian Deep Learning for Spatial Interpolation in the Presence of Auxiliary Information

Charlie Kirkwood[1] · Theo Economou[1,2] · Nicolas Pugeault[3] · Henry Odbert[4]

Received: 13 October 2020 / Accepted: 21 November 2021

Abstract Earth scientists increasingly deal with 'big data'. For spatial interpolation tasks, variants of kriging have long been regarded as the established geostatistical methods. However, kriging and its variants (such as regression kriging, in which auxiliary variables or derivatives of these are included as covariates) are relatively restrictive models and lack capabilities provided by deep neural networks. Principal among these is feature learning: the ability to learn filters to recognise task-relevant patterns in gridded data such as images. Here, we demonstrate the power of feature learning in a geostatistical context by showing how deep neural networks can automatically learn the complex high-order patterns by which point-sampled target variables relate to gridded auxiliary variables (such as those provided by remote sensing) and in doing so produce detailed maps. In order to cater for the needs of decision makers who require well-calibrated probabilities, we also demonstrate how both aleatoric and epistemic uncertainty can be quantified in our deep learning approach via a Bayesian approximation known as Monte Carlo dropout. In our example, we produce a national-scale probabilistic geochemical map from point-sampled observations with auxiliary data provided by a terrain elevation grid. By combining location information with automatically learned terrain derivatives, our deep learning approach achieves an excellent coefficient of determination ($R^2 = 0.74$) and near-perfect probabilistic calibration on held-out test data. Our results indicate the suitability of Bayesian deep learning and

✉ Charlie Kirkwood
c.kirkwood@exeter.ac.uk

[1] Department of Mathematics, University of Exeter, Exeter, UK

[2] Climate and Atmosphere Research Centre, The Cyprus Institute, Nicosia, Cyprus

[3] School of Computing Science, University of Glasgow, Glasgow, UK

[4] Met Office, Exeter, UK

its feature-learning capabilities for large-scale geostatistical applications where uncertainty matters.

Keywords Neural networks · Geostatistics · Feature learning · Uncertainty quantification · Machine learning · Mapping

1 Introduction

Maps are important for our understanding of Earth and its processes, but it is generally the case that we are unable to directly observe the variables we are interested in at every point in space. For this reason we must use models to fill in the gaps. In order to support decision making under uncertainty, statistical models are desirable (Berger 1985). Kriging—the original geostatistical model—provides smooth interpolations between point observations based on the spatial autocorrelation of a target variable (Cressie 1990; Stein 1999). However, additional sources of information are often available, thanks in part to the rise of remote sensing (Mulder et al. 2011; Colomina and Molina 2014), which provides grids of what we consider here to be auxiliary variables (e.g., terrain elevation, spectral imagery, subsurface geophysics). These are complete maps of variables that we are not directly interested in but which are likely to contain information relating to our variables of interest.

How best to extract information from auxiliary variable grids for geostatistical modelling tasks has remained an open question, but has often involved trial-and-error experimentation using manually designed filters to extract features with as much explanatory power as possible (e.g., Ruiz-Arias et al. 2011; Poggio et al. 2013; Parmentier et al. 2014; Shamsipour et al. 2014; Kirkwood et al. 2016; Kirkwood 2016; Young et al. 2018; Lamichhane et al. 2019). For example, Youssef et al. (2016) use slope angle derived from a digital terrain model as a feature to explain landslide susceptibility, but many more complex features may be useful, and these are not necessarily known in advance. To enable the utilisation of complex and unknown features, here we present an end-to-end geostatistical modelling framework using Bayesian deep learning, which frames the information extraction problem as an optimisation problem (Shwartz-Ziv and Tishby 2017), and in doing so eliminates the need for manual feature engineering and feature selection steps. Our approach therefore has the potential to supersede traditional geostatistical approaches by bringing automatic feature learning to probabilistic geospatial modelling tasks.

Here we present a two-branch deep neural network architecture—convolutional layers for feature learning combined with fully connected layers for smooth interpolation—that brings the benefits of deep learning to geostatistical applications, and we do so without sacrificing uncertainty estimation: Our approach estimates both aleatoric and epistemic uncertainties (via Monte Carlo dropout; Gal and Ghahramani 2016) in order to provide a theoretically grounded predictive distribution as output, which is composed of spatially coherent realisations (see Appendix A: Simulation). Our work brings together ideas from the fields of machine learning (Krizhevsky et al. 2012; Srivastava et al. 2014), remote sensing (Zhang et al. 2016; Zhu et al. 2017) and Bayesian geostatistics (Handcock and Stein 1993; Pilz and Spöck 2008), and unites them in a general framework for solving ‘big data’ geostatistical modelling tasks in which gridded auxiliary variables are available to support the interpolation of point-sampled target variables. We demonstrate our approach on a national-scale geochemical mapping task with encouraging results, both in terms of deterministic and probabilistic performance on held-out test data. As far as we are aware, our framework is the first to provide both well-calibrated probabilistic output and automated feature learning in the context of spatial interpolation tasks. By using neural networks, we also ensure that our framework is scalable to the largest of problems.

While the framework we present here is new, it can also be seen as a unification and generalisation of a range of prior works. Deep learning (LeCun et al. 2015)—machine learning using ‘deep’ neural networks consisting of multiple stacked layers capable of learning hierarchical composite functions—has seen increasing uptake within scientific communities in the last decade. Deep neural networks typically consist of two components: a (deep) sequence of convolutional layers designed to extract a hierarchically more efficient encoding of the signal, followed by fully connected layers at the end to estimate the desired function from the encoded representation. Both parts are trained jointly using stochastic gradient descent, ensuring that the learned features are optimised for the task. The popularity of deep learning has followed from breakthrough work by Krizhevsky et al. (2012) who achieved a new state of the art in image classification by using deep neural networks to automatically learn informative features from images (rather than manually engineering them). Deep learning has since been widely adopted within the remote sensing community (e.g., Zhang et al. 2016; Zhu et al. 2017; Li et al. 2017; Zuo et al. 2019) and has been applied to a variety of problems in geoscience, for example detecting and locating earthquakes (Perol et al. 2018), detecting faults in 3D seismic data (Wu et al. 2019) and classifying lithologies from drill core images (Alzubaidi et al. 2021). However, difficulty in obtaining reliable uncertainty estimates from deep neural networks (Kendall and Gal 2017) has meant that deep learning has not been widely adopted for applications where uncertainty matters (or, as in the aforementioned works, the proposed approaches skirt around the ever-present issue of uncertainty and simply use fixed-weight deterministic neural networks instead).

A few authors have made use of deep learning to automate feature learning in a geostatistical context (Padarian et al. 2019; Wadoux et al. 2019; Wadoux 2019; Kirkwood 2020), mostly for digital soil mapping, but only one—(Wadoux 2019)—has been able to provide uncertainty estimates, though these were achieved via a bootstrapping approach and found to be underdispersive. Here we make use of a theoretically grounded and practically effective approach to uncertainty estimation in deep neural networks: Monte Carlo dropout as a Bayesian approximation, as conceived by Gal and Ghahramani (2016). The authors are aware of one prior instance of its use in a geospatial setting: for a semantic segmentation task by Kampffmeyer et al. (2016). While our work shares similarities with the work of Kampffmeyer et al. (2016), our motivation is from the angle of geostatistical tasks in which the challenge is to utilise auxiliary information to interpolate between sparsely sampled point observations, whereas Kampffmeyer et al. (2016) tackle the remote sensing challenge of semantic segmentation: classifying each pixel of airborne images of the urban environment with their corresponding object class (e.g., car, building, tree) by training on fully manually labelled images without gaps. Both approaches require learning features from gridded data, but only ours combines this with general spatial interpolation abilities in order to provide a viable solution to the task of spatial interpolation in the presence of auxiliary information. Overall, while various separate concepts behind our work may be familiar to some readers already, here we bring them together and present Bayesian deep learning as a general solution for big data geostatistics.

Outside of the relatively recent influences of deep learning, there have also been longstanding works in the geoscience community to utilise Bayesian inference within general geological modelling practices. This initially stemmed from developments in geophysics (Tarantola and Valette 1982; Mosegaard and Tarantola 1995; Sambridge and Mosegaard 2002) in which Monte Carlo methods were presented as a means to deal with the uncertainty that is inherent to under-determined inverse problems (where many different solutions are capable of generating the observed data, i.e. the solution is non-unique—a common occurrence within the geosciences). More recently, Varga and Wellmann (2016) focused on how the Bayesian framework can be used to combine both geological knowledge and quantitative data into geological models, a theme that is further developed by Wellmann et al. (2018), Grose et al. (2019), Schaaf et al. (2020) and Olierook et al. (2021).

Our Bayesian deep learning approach for spatial interpolation in the presence of auxiliary information borrows more heavily from the machine learning and geostatistics literature than from these geological modelling works, but we acknowledge the background on both sides because we share the same motivations: the desire to incorporate all sources of information into our models, and also to characterise both aleatoric and epistemic uncertainties, such that our models will be maximally informative while remaining honest about uncertainty. Our Bayesian deep learning approach could be seen as the 'data-rich, prior knowledge-poor' end-member on a spectrum of Bayesian modelling methods, with the above-mentioned geological modelling approaches falling closer to the 'knowledge-rich, data-poor' end of the spectrum. The data-rich setting brings its own set of challenges in terms of scalability, and the need to deal with large volumes of data is a strong justification for adopting a neural network-based approach such as the one we present here.

2 Method

2.1 Feature Learning for Geostatistics

The core domain of geostatistics has been in the spatial interpolation of point observations in order to produce continuous maps in two or three dimensions. Kriging, the now ubiquitous geostatistical technique conceived by South African mining engineer Danie (Krige 1951), originally accounted for only the location and spatial autocorrelation of observations in order to produce smooth interpolations (or threshold-classified smooth interpolations in the case of indicator kriging) that can be considered optimal if no other information is available (Matheron 1962; Cressie 1990), often under assumptions of stationarity and isotropy. When other information is available, as is commonly the case today, the pursuit of optimal spatial interpolation becomes more complex. An extension of ordinary kriging, regression kriging (which is also mathematically equivalent to universal kriging and kriging with external drift; Hengl et al. 2007), allows covariates to be included in the model: the mean of the interpolated output is able to vary as a linear function of the value of covariates at the corresponding location (Gotway and Hartford 1996). For an illustrative example, the inclusion of elevation as a covariate in an interpolation of surface air temperature data could be expected to result in a map that reflects the underlying elevation map, i.e. whose mean function is a linear function of elevation. However, this quickly brings us to the limits of regression kriging: What if a linear function of elevation does not provide as much explanatory power to surface air temperature as some non-linear function of elevation? What non-linear function of elevation would be the optimal one? At the same time, what if we also have wind direction available to use as a covariate? Would we not expect the best predictor of surface air temperature to account for not just elevation, or wind direction, but how they interact, with air flowing down from mountains expected to be cooler? We quickly find ourselves in the realms of feature engineering and feature selection, a world of hypothesising and trial-and-error experimentation which has become a necessary but impractical step in the traditional geostatistical modelling process.

The defining strength of deep neural networks is their ability to learn features for themselves owing to their hierarchical structure in which the output of each layer (with a non-linear activation function applied) provides the input to the next. Through back-propagation of error gradients, neural networks can automatically learn non-linear transformations of input variables and their interactions as necessary in order to minimise a loss function. It has also been shown that in the limit of infinite width (infinite number of nodes), a neural network layer becomes mathematically equivalent to a Gaussian process (Neal 1996), which is itself the same smooth interpolator conceived by Danie Krige (i.e. kriging) under a different name. The deep neural network approach we present here combines these spatial abilities with a unique ability to learn its own features from auxiliary variable grids. We achieve this efficiently through the use of convolutional layers: trainable filters which pass over gridded data to derive new features, in a similar manner to how edge detection filters derive edges from photographs (Chen et al. 2017). By stacking convolutional layers, the complexity and scale of features that can be derived increases, along with the size of the receptive field of the neural network (Luo et al. 2016), which allows longer-range dependence structures to be learned.

2.2 Neural Network Architecture

As is shown in Fig. 1, our neural network, which we constructed and trained using Tensorflow (Abadi et al. 2016) and Tensorflow Probability (Dillonet al. 2017), has two separate input branches: a five-layer convolutional branch that takes auxiliary variable images as input (the auxiliary information branch); and a single-layer fully connected branch that takes location variables as input (the geographic location branch). The outputs of these two branches are flattened and concatenated into a single 2048 dimensional vector (128 from the convolutional branch, 1920 from the fully connected

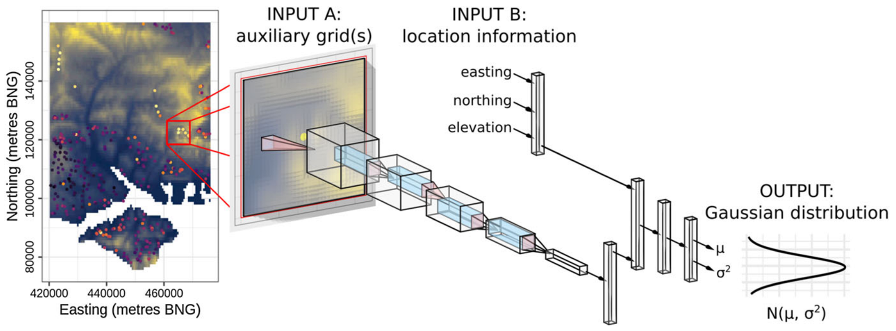

Fig. 1 Overview of our deep neural network architecture visualised with the help of NN-SVG software (LeNail 2019). For each observation, input A feeds an image of surrounding terrain into a stack of convolutional layers (shown as horizontal blocks). Simultaneously, input B feeds the observation's location variables into a fully connected layer. These two branches of the network are then concatenated and fed through a further two fully connected layers (shown as vertical blocks) from which the two parameters of a Gaussian distribution are output

branch) that feeds into the final layers of our neural network, which consist of a further two fully connected layers (1024 nodes and 256 nodes) before outputting the two parameters—mean, μ, and variance, σ^2—of a Gaussian distribution that represents the target variable. Throughout the network, we use the rectified linear unit (ReLU) activation function. In total, our neural network architecture has 2.8 million trainable parameters.

In our auxiliary information branch, the first four layers are convolutional layers, each with 128 channels, using 3x3 kernels with a stride of 1 (apart from the first, which uses a stride of 3 in order to rapidly reduce the spatial dimensions of the feature learning branch from the 32x32 input image to 10x10 and therefore reduce the number of parameters in subsequent convolutional layers). The fifth layer of the branch is a global average pooling layer, which reduces the final convolutional layer's 4x4 output into a 1x1 output by simply taking its mean. Pooling introduces translation invariance into convolutional neural networks, because the exact position at which kernels are activated is lost in the average. While it may seem counter-intuitive to instill any level of translation invariance into a spatial mapping problem, we found that it helps to reduce overfitting in the network, perhaps by encouraging it to learn features that are more descriptive of general setting (e.g. rock types) rather than of the exact location (e.g. where one specific stream meets another), and which are therefore more useful for generalising to unseen locations. We also found that average pooling outperformed max pooling for this use case, presumably because it produces smooth transitions between the contextual features present at adjacent locations (rather than having features pop in and out depending on whether they are present anywhere within the extent of the auxiliary information image). While the convolutional architecture we propose here is effective, we also suspect that there is room for improvement, and would encourage further research in this area.

The geographical location branch (from input B in Fig. 1) of our neural network is simpler, and consists of a single fully connected layer whose output is concatenated with the output of the auxiliary information branch before feeding into a further two fully connected layers prior to the neural network's output. Without the inclusion of the auxiliary information branch, our model would simply be a deep fully connected neural network operating on spatial location inputs (easting, northing, elevation), and could therefore be regarded as approximately performing 'deep kriging', or spatial interpolation using deep Gaussian process regression, which is what our neural network would become in the limit of infinite layer width (Neal 1996). The idea of 'deep kriging' has been explored by Li et al. (2020) in their paper of the same name, who propose a neural network architecture that uses an embedding layer to transform the input space. The inclusion of the auxiliary information branch in our neural network, however, turns it into something closer to 'deep regression kriging', a term that seems not to have previously been coined. Unlike traditional regression kriging, not only does our deep neural network architecture learn its own regression features automatically from gridded auxiliary information, but also learns interactions between these features and spatial location owing to the fact that we have subsequent hidden layers after the auxiliary and spatial information branches are concatenated. We can therefore think of our neural network as performing interpolation in a self-learned hybrid space—a representation which blends global location information with local contextual information.

Therefore, our neural network provides a forward model from the location information and auxiliary information inputs, whose output is a Gaussian distribution representing the target variable (stream sediment calcium concentrations in the example we present here) via two parameters: the mean, μ, and variance, σ^2. These two output parameters are generated by separate linear functions of the same 256 learned features that constitute the neural network's final hidden layer (which in this case has 256 nodes). While μ is free to vary on the real line, we constrain σ^2 to always be positive (and therefore valid) by using a softplus link function: $\text{softplus}(x) = \log(\exp(x) + 1)$. Because these output parameters have access to the same features, there may be some amount of cross-talk between them. This could be forcibly avoided by using separate neural networks to learn each of these output parameters. However, given the nature of the problem, it is likely that both μ and σ^2 will vary according to the same underlying processes, such as changes in lithological or hydrological setting. It therefore seems a reasonable approach to allow both μ and σ^2 the capacity to share the same features (and to have them jointly inform the learning of these features). The quality of our results empirically supports this reasoning.

Our probability model $p(Y_s|x_s, w)$ for our target variable Y_s, at any location s, given the inputs x_s and weights of the neural network w is defined as

$$\begin{aligned} Y_s|x_s, w &\sim \mathcal{N}(\mu(x_s),\ \sigma(x_s)^2) \\ \mu(x_s) &= g(layer_{final}) \\ \sigma(x_s)^2 &= \log(\exp(h(layer_{final})) + 1), \end{aligned} \tag{1}$$

where both g and h are linear functions of the 256 features of the final hidden layer of the neural network ($layer_{final}$) so that both take the form $g(layer_{final}) = \beta_0 + \beta_1 feature_1 + \beta_2 feature_2 + \beta_3 feature_3 + \cdots + \beta_{256} feature_{256}$. Each of these 256 features is itself a learned transformation of the neural network's inputs, x_s, namely location information and auxiliary information (Fig. 1). The specifics of these functions are dictated by the trainable parameters, or weights, in the neural network. In the next section, we explain how these parameters, and the functions they induce the neural network to represent, are learned.

2.3 Quantifying Uncertainties via Monte Carlo Dropout

Traditionally, neural networks are deterministic models in that they provide a fixed output for a given input, subject to the values of their parameters (or weights, w). Neural networks are commonly trained through back-propagation to converge on a set of weights that minimise a loss function (often mean squared error in the case of regression problems). However, in these traditional deterministic neural networks the weights are fixed, having no distribution, which means that there is no way to estimate the uncertainty in these weights or therefore the uncertainty about the function or model that the neural network has learned. Natural processes inevitably involve uncertainties, and it is right that we should want to estimate these in order to provide well-calibrated probabilistic predictions suitable for use in decision support (Yoe 2011; Fox 2011). We do so here using the Monte Carlo dropout approach of Gal and Ghahramani (2016) for approximate Bayesian inference, which allows us to capture both aleatoric and epistemic uncertainty as described by Kendall and Gal (2017). Aleatoric uncertainty can be thought of as the innate randomness in a data-generating process—irreducible noise inherent in the observations of the target variable—and can be represented by using a parametric distribution as the output of the neural network so that, rather than making single point predictions, our model predicts a distribution whose variance acknowledges the inherent randomness in the observations. In our case, our deep neural network outputs the mean, $\mu(x_s)$, and variance, $\sigma(x_s)^2$, of a Gaussian distribution (Eq. 1) which provides our likelihood function for the neural network ($\mathcal{L}_{NN}$). If we define vector $\boldsymbol{y} = (y_1, \ldots, y_n)$ to be the observed data on Y_s (our output variable), then the likelihood is defined as the joint probability of the data $\boldsymbol{y}$ given specific values of $\mu(x_s)$, $\sigma(x_s)^2$ and w. This is given by

$$\begin{aligned}\mathcal{L}_{NN} &= p(\boldsymbol{y}|\mu(x_s), \sigma(x_s)^2, w) \\ &= \prod_{s=1}^{n} p(y_s|\mu(x_s), \sigma(x_s)^2, w) \\ &= \prod_{s=1}^{n} \frac{1}{\sqrt{2\pi\sigma(x_s)^2}} \exp\left(-\frac{1}{2\sigma(x_s)^2}(y_s - \mu(x_s))^2\right) \\ &= \frac{1}{(2\pi\sigma(x_s)^2)^{n/2}} \exp\left(-\frac{1}{2\sigma(x_s)_s^2} \sum_{s=1}^{n}(y_s - \mu(x_s))^2\right).\end{aligned} \tag{2}$$

If we were not also interested in epistemic uncertainty—uncertainty within the model itself—we could simply optimise the weights, w, to arrive at a fixed set which maximises the likelihood (maximises the probability of the data given the model). Assuming a Gaussian error distribution as we do here, maximising the likelihood would lead our neural network to fit the predictive mean, $\mu(x_s)$, equivalently to if we were minimising the mean squared error (MSE),

$$MSE = \sum_{s=1}^{n} (y_s - \mu(x_s))^2, \tag{3}$$

which is perhaps the most commonly used loss function for deterministic regression problems. However, by having our neural network learn both the mean, $\mu(x_s)$, and the variance, $\sigma(x_s)^2$, of our target variable as functions of the inputs, we can learn a spatially precise heteroscedastic (Kendall and Gal 2017) representation of the uncertainty within the data—the aleatoric uncertainty.

In addition to modelling the aleatoric uncertainty, we also wish to model the epistemic uncertainty—the uncertainty within the model itself. We do so in acknowledgement of the fact that, given that we do not have complete and perfect information (our set of observations is finite), there is uncertainty about the form of the true data-generating process. This uncertainty can be reduced by collecting more data, but without infinite observations there will always be room for multiple possible model fits, or explanations, for the data we observe. If we simply task our neural network with learning a single 'best fit' function to represent the data-generating process (e.g. by maximising the likelihood), then we would be ignoring this epistemic uncertainty about the range of possible fits, resulting in overconfident predictions and poor generalisation. Instead, to model the epistemic uncertainty, we model a distribution over the range of possible model fits. To do so requires operating within the Bayesian framework to learn a distribution over the neural network weights, rather than simply learning a fixed set of weights as in traditional frequentist neural networks. However, learning a distribution over each weight, or parameter, in the model is a challenging proposition for large neural networks due to the extreme dimensionality of the model (2.8 million trainable parameters in our case).

Here we use the Monte Carlo dropout approach for approximate Bayesian inference in deep neural networks (Gal and Ghahramani 2016). This approach places a Bernoulli prior row-wise over the weight matrices of the neural network, which means that for every iteration of training and prediction, the nodes of the neural network each have a probability of being switched off, or 'dropped out' (with weights set to zero). The probability or rate at which nodes will drop out is a tuneable hyper-parameter. While a Bernoulli prior may seem 'unrealistic'—why should a parameter only exist with a fixed probability?—the overall effect of Monte Carlo dropout on the network as a whole is to turn our single neural network into a near-infinite self-contained ensemble. Each different configuration of dropped nodes realises a different function (or model) from the ensemble, so that rather than learning a single 'best' fit, our neural network learns a distribution over possible fits.

The dropout rate relates to the variance we expect to see between different functions drawn from the ensemble—it acts as our prior distribution over functions. In general, a higher dropout rate will induce higher variance within the ensemble, as samples (or 'ensemble members') become less correlated. However, the dropout rate also affects the capacity of the neural network to represent complex functions; for example, a high dropout rate of 0.9 would on average leave only 10% of the nodes of the network active for any given sample, effectively causing the ensemble to be composed of much smaller neural networks which would likely be weaker learners (Srivastava et al. 2014). In the same way that theory does not dictate the optimal neural network architecture for a given task, so too the optimal dropout rate is task- (and architecture-)dependent. By manually tuning the dropout rate in order to minimise loss on the evaluation set, a well-calibrated posterior predictive distribution can be achieved. For our geochemical mapping application, we found a dropout rate of 0.2 for the fully connected layers, and a spatial dropout rate of 0.5 for the convolutional layers (in which filters, rather than nodes, are dropped) to be effective. The manual tuning of dropout rates adds some time to the model development process, but it is just one of many hyper-parameters to consider in the design of the neural network (along with depth, layer width, activation function, convolutional kernel size, dilation, stride, pooling, etc.). It is worth noting that other approaches to Bayesian inference in deep neural networks have been proposed, in what is a rapidly developing area of research [we recommend the concise review by Wilson (2020)]. We therefore remain open-minded about what may emerge as the 'best' approach over the coming years, but have found Monte Carlo dropout to perform well in our task.

We now provide a deeper look at the principles behind Monte Carlo dropout as a Bayesian approximation, though for the complete details, we refer readers to the original work of Gal and Ghahramani (2016). First, consider the simpler scenario of using our neural network without dropout. In this scenario, the gradient descent training procedure aims to find a fixed set of weights, $w*$, which maximise the likelihood: the probability of the data given the weights, $p(\boldsymbol{y}|w*)$. Given its enormous number of parameters, our neural network could potentially achieve this by fitting the mean of its Gaussian output, μ, directly through our training observations, and setting its variance, σ^2, close to zero everywhere—although this would undoubtedly be a case of overfitting the data. Regularisation techniques can be used to prevent this overfitting by penalising complexity, but regardless, the outcome would still be a fixed set of weights, $w*$, which provides no estimate of epistemic uncertainty.

In order to quantify epistemic uncertainty, we want to learn the posterior distribution of the weights, $p(w|\boldsymbol{y})$, given the data $\boldsymbol{y}$. To do so in the traditional way requires combining the likelihood, $p(\boldsymbol{y}|w)$, with a prior distribution for the weights, $p(w)$, through Bayes' rule. In our case, our prior is constructed by assuming randomly initialised fixed weights, β, and for each node of the network (each row of the neural network's weight matrices) a Bernoulli random variable, $z \sim \text{Bern}(\pi)$, where $\pi = \Pr(z = 1)$. Then the distribution over weights is defined as $p(w) = \beta z$, which defines whether the weight β is 'active' ($z = 1$) with probability π or 'dropped out' ($z = 0$) with probability $1 - \pi$, where $1 - \pi$ is the dropout rate to be tuned manually as a hyper-parameter. Using Bayes' rule, the posterior is defined as

$$p(w|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|w)p(w)}{p(\boldsymbol{y})}, \tag{4}$$

but the Monte Carlo dropout approach provides an approximation whereby we obtain the posterior by training the fixed weights, β, while dropout is active at the rate we specify. Once training is complete, we take the Monte Carlo dropout distribution, $\beta_{TRAINED}z$, to be our posterior, so that

$$p(w|\boldsymbol{y}) = \beta_{TRAINED}z. \tag{5}$$

This approximation is efficient in that, once the fixed weights have been trained (for which we can use the standard efficient optimisers for deep learning), the Monte Carlo dropout samples immediately provide independent samples from the posterior, $p(w|\boldsymbol{y})$, with no burn-in or thinning required, unlike samples obtained by Markov chain Monte Carlo (MCMC) methods [e.g. see Raftery and Lewis (1996)]. However, it does mean that we are entirely dependent on learning an optimal set of fixed weights, $\beta_{TRAINED}$ (subject to the dropout rates we specify), in order that our approximate posterior results in a well-calibrated model. From the perspective of big data efficiency, this is in fact a selling point of the Monte Carlo dropout approach, as it means we can use the established tools for training deep neural networks very efficiently, namely stochastic gradient descent in frameworks such as Tensorflow, and treat the dropout rate as a hyper-parameter to be tuned in the neural network design process.

To arrive at our trained weights, $\beta_{TRAINED}$, in Tensorflow, we task stochastic gradient descent with optimising β to minimise the negative log-likelihood loss, $-log[p(\boldsymbol{y}|w)]$, which equates to maximising the likelihood. However, with dropout active, it is no longer sufficient for the optimiser to find a single point in parameter space that provides maximal likelihood (which would likely be a sharply peaked maximum corresponding to a model that fits perfectly through the training data but generalises poorly to new data). Instead, the optimiser must find a maximum that can accommodate the dispersion induced by the activity of Monte Carlo dropout. The effect of this is that the mean of the resultant posterior is regularised compared to the maximum likelihood estimate without Monte Carlo dropout active. In addition, the posterior, being an optimised dropout-induced distribution over the weights given the data, provides an estimate of the epistemic uncertainty in the model.

The uncertainty estimates obtained from Bayesian methods will always have some sensitivity to the choice of prior. In our case, the dispersion of our Monte Carlo dropout posterior is sensitive to our choice of the dropout rate. Fortunately, in a big data setting such as ours, we can use a large evaluation data set to help tune the dropout rate and any other hyper-parameters (more details of our specific setup follow in the next subsection). Our aim with tuning the dropout rate is that the stochastic gradient descent training process should arrive at a solution $\beta_{TRAINED}$ which corresponds to a model that fits the training data as closely as possible without overfitting, which would present itself as a degradation of predictive performance on the evaluation set. We can see this as it happens by monitoring the training and evaluation loss during the stochastic gradient descent process in Tensorflow. With dropout at a suitable rate, training loss will continue to decrease as stochastic gradient descent continues, while evaluation loss will reach a low plateau and stay there. This indicates that the resultant posterior corresponds to a model that well represents both the training data and the evaluation data, without becoming overconfident (i.e. overfitting). The better the uncertainty quantification, the better the predictive performance will be outside of the training data.

The posterior distribution is the key to capturing epistemic uncertainty—it represents the uncertainty in the function, or model, that the neural network has learned. Given the posterior distribution over weights, $p(w|\boldsymbol{y})$, the posterior predictive distribution of any output Y_s given the associated value of the input x_s is given by

$$p(Y_s|x_s, \boldsymbol{y}) = \int_w p(Y_s|x_s, w)p(w|\boldsymbol{y})\mathrm{d}w, \tag{6}$$

where $p(Y_s|x_s, w)$ is our Gaussian model (Eq. 1). Note that Y_s in Eq. 6 can be any value of the output variable, observed and unobserved alike; e.g. Y_s can be a location s not covered by the observed data $\boldsymbol{y}$. In practice, we calculate this integral using Monte Carlo integration: simulating from the posterior predictive distribution for a given input, x_s, one sample at a time, where each sample is drawn from the Gaussian distribution using a different arrangement of weights w_i, sampled (by Monte Carlo dropout) from the posterior distribution $p(w|\boldsymbol{y})$. For more on simulation, and the spatial properties of resultant realisations, see Appendix A.

2.4 Application to Geochemical Mapping

We applied our Bayesian deep neural network to the task of mapping stream sediment calcium concentrations, as log(calcium oxide), across the UK. This geochemical data set, provided by the British Geological Survey, contains 109,201 point-sampled calcium observations (as well as many other elements) measured by chemical assay of sediment collected from the beds of streams across the UK, approximately at random (Johnson et al. 2005). For our auxiliary grid, we use NASA's Shuttle Radar Topography Mission (SRTM) elevation data (Van Zyl 2001), which we access via the Raster package in R (Hijmans 2017; R Core Team 2020) at a resolution of 30 arc-seconds, which translates to a horizontal resolution of 528 m and a vertical resolution of 927 m once projected into the British National Grid coordinate system. In total, for the UK this provides 611,404 grid cell elevation values. We chose calcium concentration partly for its ability to differentiate rock types: calcium carbonate is the main constituent of chalk and a major constituent of limestones, but can be almost completely absent from deeper marine sediments deposited below the calcite compensation depth. We also chose calcium for how easily weathered and mobile it tends to be in the surface environment, which means that it exhibits a complex relationship with terrain topography. Not only can we expect terrain features to be indicative of underlying bedrock composition (due to different rock compositions weathering differently, producing different surface expressions), but mobile elements will also be transported according to hydrological processes at the surface. In order to make good predictions of calcium concentrations, our neural network therefore has to learn and combine knowledge of both bedrock and surface processes.

Constructing our study data set required linking together the two input types (location information, and auxiliary information in the form of an observation-centred terrain image) to each observation of our target variable. Location information consists of the easting and northing values recorded for each observation in the G-BASE data set, along with an elevation value extracted from the SRTM elevation grid at that location. The observation-centred terrain images each consist of 1,024 elevation values extracted on a regular 32x32 grid centred at the location of the observation site. Bilinear interpolation of the elevation grid was used in all elevation extractions in order to avoid aliasing issues in the terrain images. We extract the terrain images at a grid cell size of 250 m, which means that the neural network has an 8x8-km square window centred on each observation from which to learn its terrain features. Constructing the auxiliary information images through bilinear interpolation also means that we are not tied to the resolution of the underlying auxiliary grids. It is worth noting that our framework is capable of ingesting multiple auxiliary variable grids at once (multi-channel images as input), and there is no obligation to use only terrain. For instance, other sources of auxiliary information such as satellite or geophysical imagery may also be available.

In order to facilitate the learning of terrain features, we normalise each 32x32 cell image so that the centre point is at 0 elevation. Features are then learned in terms of contextual relation to the sample site, rather than to absolute elevation. However absolute elevation, along with easting and northing, are provided explicitly as the second input to the neural network (after the convolutional layers—see Fig. 1) in order to provide the network with awareness of overall location in the geographic space as well as awareness of local topography (i.e. from the auxiliary information). All location inputs are scaled to have a mean of 0 and standard deviation of 1, with the elevation images also collectively scaled to have standard deviation of 1, but without further centring beyond setting the centre of each image to 0 elevation.

In order to conduct our study, we split our assembled data set at random into ten folds, of which one was set aside as a final test data set (from which we report our prediction accuracy and calibration results in Sect. 3), one was used as an evaluation set during the neural network training (to monitor loss on out-of-sample data to guide hyper-parameter tuning, such as tuning of dropout rates), and the remaining eight folds were used as the training set (which amounted to 87,361 training observations). Different proportions could have been chosen for this hold-out validation scheme, but we have chosen tenths on the basis that this is common practice and that our held-out test set is sufficiently large (10,920 observations) for us to be confident in our results. To train the neural network, we used the Adam optimiser (Kingma and Ba 2014) with a learning rate of 0.001, weight decay of 1e-6 and a batch size of 4,096. With the dropout rate tuned to a suitable value (we settled on 0.2 for the fully connected layers and 0.5 for the convolutional layers), we found that our neural network was resistant to overfitting even when trained for a large number of epochs. This can be seen during training by monitoring predictive performance on the evaluation set, which plateaued after many epochs but did not degrade. This is a good sign, as it suggests that the posterior predictive distribution had become a good approximation of the true data distribution. We trained our neural network for 1,000 epochs, which took about 25 minutes on a single GPU workstation (Nvidia Pascal Titan X GPU). Note that all of our result metrics are reported from the third data set—the test data set—which was not used during training at all. We would therefore expect the results we present in Sect. 3 to well represent the general performance of our method in the context of predicting values of log(CaO) at unobserved locations within the UK.

We chose the SRTM and G-BASE data sets for their ease of access and use as well as for the complexity of the spatial relationships they contain, which we believe provide a good demonstration of the capability of our Bayesian deep learning approach for geostatistical modelling tasks. The methodology we present in this paper is intended as a general framework for data-rich geostatistical applications where gridded auxiliary variables are available in addition to point-sampled observations of the target variable, and we would encourage readers to use the code we share alongside this paper (available at https://github.com/charliekirkwood/deepgeostat) to test the approach on other geostatistical applications.

3 Results and Discussion

The national-scale geochemical map that our Bayesian deep neural network has produced (Fig. 2) is extremely detailed and appears to have successfully captured the complex relationships between our target variable, stream sediment calcium concentrations as log(CaO), and our auxiliary variable grid, terrain elevation. In addition to subjectively achieving good detail in the mapping task, our objective results on held-out test data (unseen during the model training and hyper-parameter tuning procedure) are very encouraging. In a deterministic sense, the mean prediction from our Bayesian deep neural network explains 74% of the variance in our target variable (Fig. 3a). The performance of the network in a probabilistic sense is less easily summarised by a single number, but a comparison of the predictive distribution with the true distribution on the held-out test set (Fig. 3b, c) indicates a well-calibrated fit (Gneiting et al. 2007). We have also measured performance using two proper scoring rules (Gneiting and Raftery 2007), the continuous rank probability score (CRPS) and logarithmic score (Fig. 3b), though these will be most useful in future comparisons with other models.

It is apparent in the observation data presented in Fig. 3 that a cluster of observations share the same identical value of -3.69 log(CaO), which is the lowest value observed and corresponds to 0.025 weight % CaO on the linear scale. We believe that this set of identically valued observations are the result of a bug in error correction of the assay data for low-calcium-concentration samples, and we are therefore not concerned by the discrepancy between our predictions and these observations (the neural network predicts that all should have higher values than recorded). It may even make sense to exclude these spurious observations from our comparisons, but we leave them in as a reminder that data sets in general are not necessarily free of defects, and that probabilistic data models like ours can in fact be a good way to identify statistically implausible defects like these.

Checking the coverage of our prediction intervals on the held-out test data (Fig. 3d), we find that 94.6%, 71.4% and 51.6% of observations fall within the 95%, 70% and 50% prediction intervals, respectively. We take these numbers (which we may expect to be slightly skewed by the above-mentioned spurious measurements in the low tail of the data) on a relatively large held-out test set (10,920 observations, 10% of the total data set) as evidence that our Bayesian deep neural network is providing well-calibrated probabilities, and therefore that it would be reasonable to use the predictive distribution to support decision making (Gneiting and Katzfuss 2014).

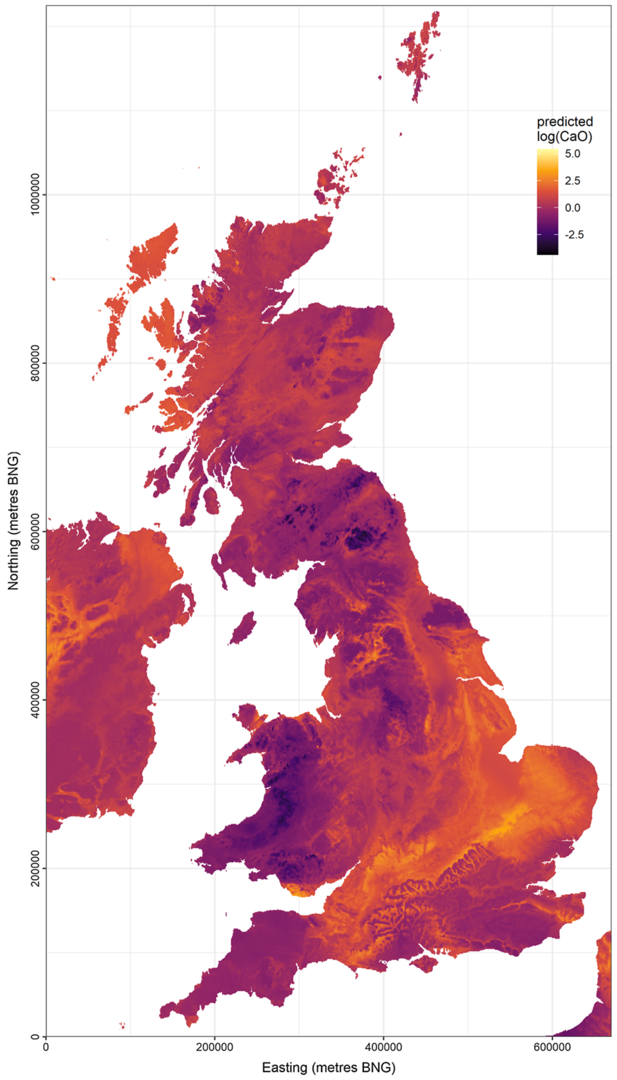

Fig. 2 Predicted log(CaO) interpolated from stream sediment geochemistry observations across the UK using auxiliary information provided by a digital elevation model. This map shows the mean of our deep neural network's predictive distribution, which has captured complex relationships between terrain features and log(CaO)

We visualise the probabilistic capabilities of our deep neural network using a south–north section line through the map along the 400,000-metre easting grid line (Fig. 4). In doing so, we can see that the neural network is able to represent epistemic and aleatoric uncertainty independently as necessary to minimise loss. The credible interval for the mean varies spatially despite the fixed rate of Monte Carlo dropout, showing that the neural network is able to capture spatial variability in epistemic uncertainty. Likewise, the estimated aleatoric uncertainty also varies spatially, and can be high even

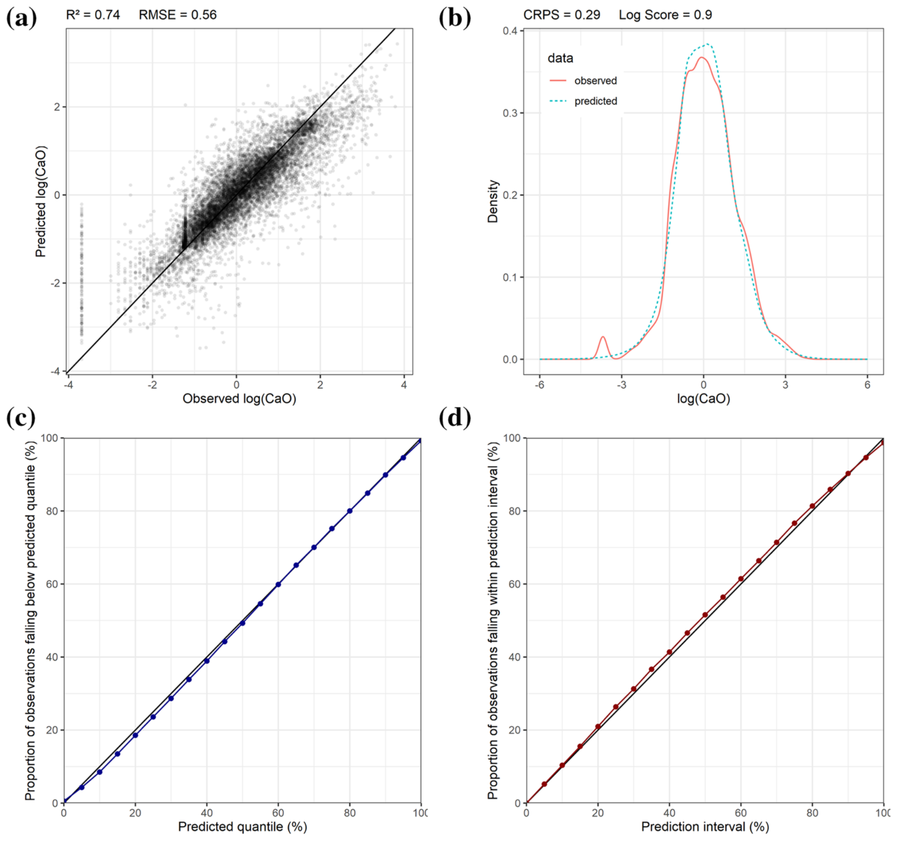

Fig. 3 Evaluating our model's performance on the held-out test data set ($n = 10,920$). **a** Comparison of observed and predicted values, taking the mean of the predictive distribution as a deterministic prediction. **b** Density, **c** Q-Q and **d** prediction interval coverage plot comparisons of observed and predicted distributions

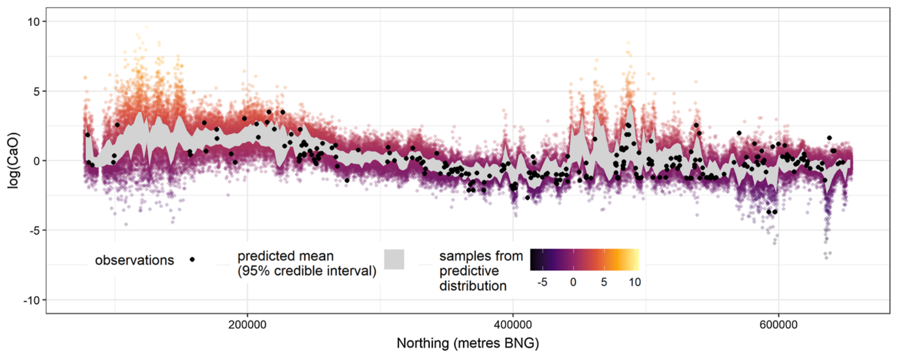

Fig. 4 South–north cross section of our Bayesian deep neural network's output, running along a line at 400,000 metres easting BNG. Also shown are all the observations within 500 m either side of this line

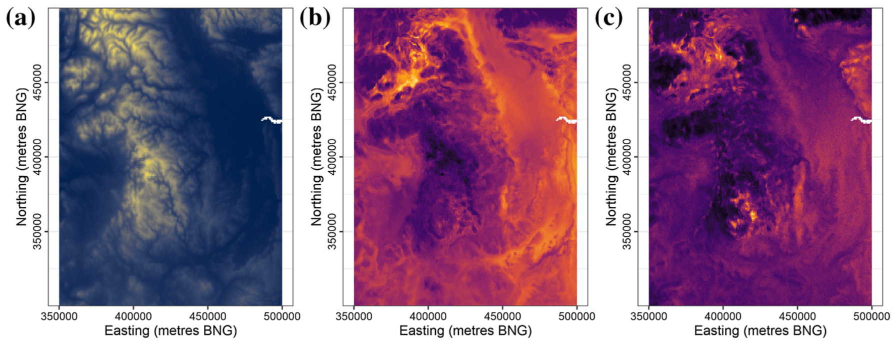

Fig. 5 Zooming in on a local area. **a** SRTM elevation data: the source of our auxiliary information. **b** Predicted log(CaO): deterministic mean. **c** Uncertainty of predicted log(CaO): standard deviation of posterior predictive distribution. All maps use linear colour scales where brighter = greater. The white inlet is the tip of the Humber estuary

where epistemic uncertainty is low. For example, we see this behaviour exhibited just south of 600,000 metres northing. A quick check of the British Geological Survey's Geology of (Britain Viewer British 2020) suggests that the geology of this section consists of the Yoredale group of interbedded limestone, argillaceous rocks and subordinate sandstone. The interplay of these compositionally different rock types on fine spatial scales that are unresolvable to the model (and to the geologists who classified the formation) is likely the reason for the comparatively high aleatoric uncertainty estimates in this area even with low epistemic uncertainty: the model has recognised that calcium concentrations here have higher variability at short spatial scales, and has increased its 'nugget' variance to account for this. This is one example of how probabilistic machine learning can be used as a guide towards discovery of further knowledge. By outputting a full predictive distribution, the Bayesian deep learning approach can provide probabilistic answers to all sorts of questions (e.g., Cawley et al. 2007; Kirkwood et al. 2021). Probabilities of exceedance at any location, for example, can be calculated simply as the proportion of probability mass in excess of any chosen threshold. We can also obtain individual realisations from the model through simulation. These realisations have spatial autocorrelation properties similar to that of the data—see Appendix A for further details.

We zoom in on the national-scale map and visualise predictive uncertainty in Fig. 5. Viewing the deterministic mean map at this finer scale, and comparing it to the elevation map of the same extent reveals in more detail the ability of our deep convolutional neural network to learn the complex ways in which the distribution of our target variable relates to features of terrain. The same level of complexity is reflected in the uncertainty map and shows that in addition to being very well calibrated (Fig. 3), our Bayesian neural network is also very specific in its assignment of uncertainty to different spatial locations. In other words, our predictive distribution is both honest and sharp, which is desirable under the paradigm proposed by Gneiting et al. (2007) that probabilistic predictions should ideally be achieved by maximising the sharpness of the predictive distribution subject to calibration. Our combination of high map detail (in terms of both predictive mean and variance) and near-perfect coverage indicates that our Bayesian deep learning approach is successfully producing a predictive distribution that is both sharp and well calibrated.

Our deep neural network is able to produce these specific and detailed outputs because it is interpolating not just in geographic space—as in traditional geostatistical models—but also in terrain feature space. This has important implications for mapping tasks. In traditional geostatistical models, any predictions made outside the geographic extent of observations would be considered to be extrapolations, and are likely to have high error and uncertainty (Journel and Rossi 1989). In our case, because our neural network is working in a hybrid space, predictions that would be considered out of sample geographically may still be within sample in terms of terrain features. While regression kriging may also be provided with terrain features as covariates, only a deep learning approach like ours has the capability to automatically learn complex terrain features for itself, and therefore has the potential to discover new ways to predict target variables based on fundamental relationships with the landscape rather than relying on spatial autocorrelation. This may have significant implications for applications like mineral exploration, where obtaining sensible predictions for unexplored regions is a key driver of new discoveries (Sabins 1999). It also has implications for sample design in the age of 'deep geostatistics', which we leave for future work, other than to say that sample design ought to consider both the geographic space and the terrain feature space, and would likely be best guided by the epistemic uncertainty estimates of the deep models themselves.

The effects of fluvial processes on calcium are perhaps the most noticeable terrain-related effects captured in the map, with downslope 'washing out' of calcium apparent in valleys. In the zoomed-in region of Fig. 5, for example, we can see elevated calcium concentrations in the channels that drain away from the calcium-rich area in the north-west of the figure (coordinates approx. 380,000, 450,000) even where these channels cross through otherwise low-calcium areas. This suggests that the convolutional branch of our neural network may have learned the concept of hydrological catchments and associated sediment transport directly from the data, a capability that Zuo et al. (2019) suggest will be important for improving robust mapping of geochemical anomalies in the future. Further work will be needed to fully explore the capacity of our approach to learn complex physical process by example, and perhaps also to investigate the physical plausibility of the resultant predictions. However, the authors are aware of no other methods that could match the capabilities of our Bayesian deep learning approach in this geochemical mapping task. Numerical models may be able to represent physical processes more accurately, but they struggle to accurately quantify uncertainties. Conversely, traditional geostatistical modelling approaches like regression kriging may do well at quantifying uncertainties, but have no capabilities in feature learning, which limits their capacity to fully utilise the information contained within auxiliary data sets. An approach known as topographic kriging (Laaha et al. 2014) has been developed specifically for interpolation on stream networks, but this is unable to generate predictions outside of the manually designated stream network, and so is of limited use for general mapping applications. We therefore postulate that the Bayesian deep learning approach we present here represents a step change in capabilities over previous geostatistical approaches, for its ability to automatically learn such complex relationships between target variables and the landscape.

4 Conclusion

Our Bayesian deep learning approach to spatial interpolation in the presence of auxiliary information achieves excellent predictive performance on held-out test data according to both probabilistic and deterministic metrics, and in doing so produces maps with a high level of functional detail whose well-calibrated probabilities would be suitable for use in decision support. Our approach is unique in combining the following capabilities: (I) automated information extraction from auxiliary variable grids via convolution, (II) pure spatial interpolation abilities not dissimilar to that of ordinary kriging (each fully connected layer in our neural network architecture would be equivalent to a Gaussian process in the limit of infinite layer width), (III) outputting a well-calibrated predictive distribution by using Monte Carlo dropout for approximate Bayesian inference and (IV) the ability to handle very large data sets, including compatibility with GPU acceleration. As such, our approach brings new feature learning abilities and 'big data' efficiencies from deep learning to the established geostatistical domain of probabilistic spatial interpolation.

The major benefit of our end-to-end deep learning approach is the ability to automatically learn and utilise the complex relationships between auxiliary grids and target variables that it would not be possible or practical to manually specify, for example capturing the effects of fluvial processes on calcium distributions in our demonstration. Traditional geostatistical methods have no ability to automatically learn features, hence the significance of this work. By improving our ability to utilise auxiliary information in mapping tasks, we also reduce reliance on spatial autocorrelation for making predictions. This has the potential to improve the generalisation of geostatistical models, including beyond the spatial extents of a study area, owing to the potential of deep learning approaches to learn the 'fundamental truths' that may relate target variables to auxiliary grids. This potential remains to be explored.

Acknowledgements We acknowledge funding from the UK's Engineering and Physical Sciences Research Council (EPSRC project ref: 2071900) and from the UK Met Office, both of whom fund Charlie Kirkwood's PhD research. Theo Economou was funded by the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no. 856612 and the Cyprus Government. We would like to thank our anonymous peer reviewers for their insightful comments which have helped to strengthen this manuscript. Our thanks also go to Nvidia and their GPU grant scheme for kindly providing Charlie Kirkwood with the GPU that trains our deep neural network in under 30 minutes, and also to the British Geological Survey for making the G-BASE geochemical data set available for this study. The code to reproduce this study is available at https://github.com/charliekirkwood/deepgeostat and includes functions to download NASA's SRTM elevation data via the raster package in R. We are unable to provide open access to the stream sediment geochemistry target variable data set; however, for academic research purposes, readers may request access to this data set from the British Geological Survey at https://www.bgs.ac.uk/enquiries/home.html or by email to enquiries@bgs.ac.uk.

Appendix A: Simulation

Our Bayesian approach means that we have not learned just a single 'best fit' for our neural network, but a distribution over possible fits (see Sect. 2.3) from which we can simulate different spatially coherent maps, or realisations (Fig. 6). Each realisation presents predictions from a possible model fit, which collectively construct our posterior predictive distribution, which itself represents our current state of knowledge. Each realisation can therefore be thought of as depicting what we might observe if the data collection procedure was repeated (at all grid cells of the map), subject to our current state of knowledge. We simulate these realisations by sampling from our posterior predictive distribution (presented previously as Eq. 6),

$$p(Y_s|x_s, \boldsymbol{y}) = \int_w p(Y_s|x_s, w)p(w|\boldsymbol{y})\mathrm{d}w. \tag{7}$$

We do so by iterating two steps. First, we sample $w_i \sim p(w|\boldsymbol{y})$, which is to say we sample one configuration of weights from our posterior distribution. Second, we sample $Y_{s_i} \sim p(Y_s|x_s, w_i)$; in other words, for each location s across the map, we sample one data value from our sampling distribution (the Gaussian output of our model) conditional on both the inputs to the neural network at that location and the configuration of weights from the first step. The form of our model—in which we represent aleatoric uncertainty using independent Gaussian noise—means that, conditional on x_s and w_i, the simulated values Y_s are independent for each spatial location s. This independent noise can also be thought of as our model's 'nugget effect' (Clark 2010).

In practice, due to the fact we are using Monte Carlo dropout, making predictions across an entire map using the same sampled configuration of neural network weights, w_i, requires freezing the dropout mask for multiple calls to Tensorflow's predict function (one call for each grid cell of the map). We have provided code to achieve this, as this functionality is not built in to Tensorflow, which, when Monte Carlo dropout is active, would normally sample a new configuration of weights for each individual prediction, preventing spatially coherent maps of posterior predictive samples (i.e. realisations) from being obtained.

Whether or not we use dropout mask freezing in order to realise spatially coherent realisations, the posterior predictive distribution at any location, $p(Y_s|x_s, \boldsymbol{y})$, remains the same. In this paper, we have focused on the quality of our deep neural network's posterior predictive distribution, as assessed by its ability to provide good probabilistic predictions at the locations of unseen held-out test data (see Fig. 3 and Sect. 3). This is the general use case that we envision our Bayesian deep learning approach being used for. However, for some applications, users may be interested in the properties of individual realisations in addition to the properties of the predictive distribution overall. For example, in resource estimation and mine planning, obtaining realisations that fit the main characteristics of the revealed reality (Journel 1974, of which spatial

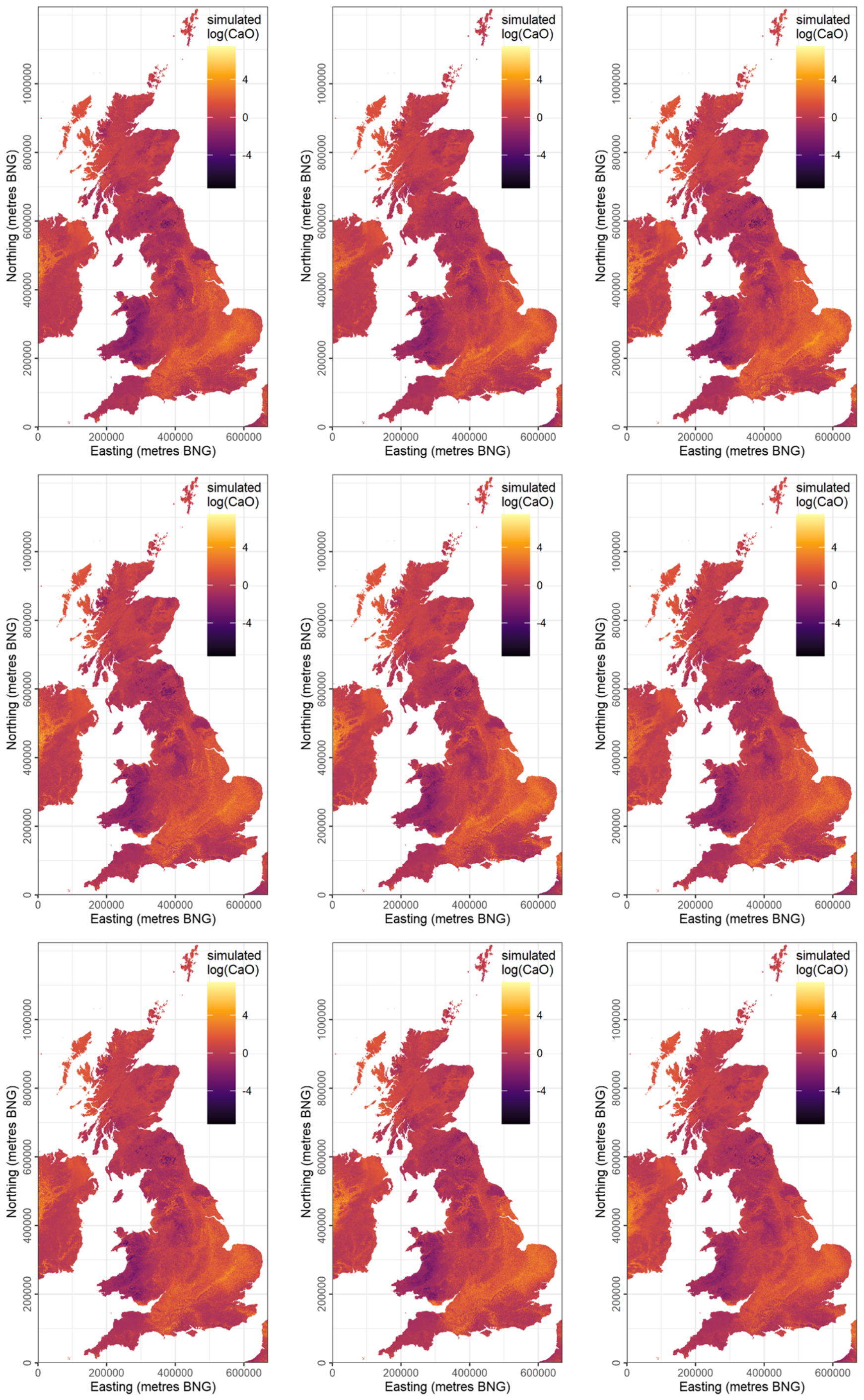

Fig. 6 Nine simulated maps, or realisations, from our deep neural network. Each map is a sample from the posterior predictive distribution. Crossing your eyes to focus on two maps at once can help to make the differences more apparent

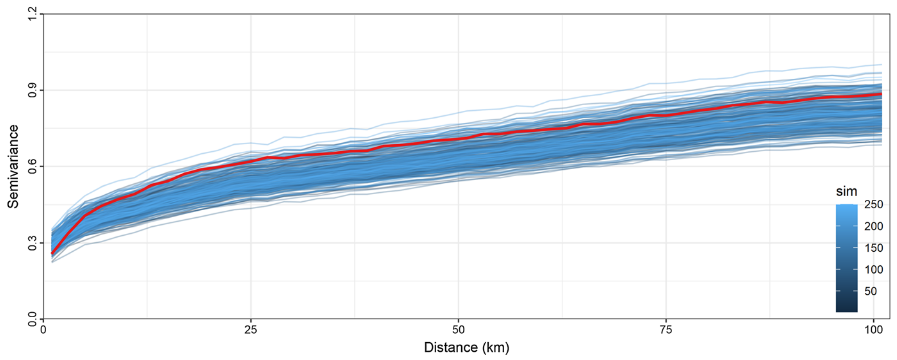

Fig. 7 In blue: semi-variograms for 250 simulations of log(CaO) values predicted at the locations of the held-out test observations ($n = 10,920$). In red: semi-variogram of the held-out test observations. Variograms produced using a bin size of 2 km

autocorrelation is seen as most important) has enabled more efficient optimisation of mining activities (Dimitrakopoulos 1998, 2018; Menabde et al. 2018) and analysis of risk (Vann et al. 2002).

Unlike traditional geostatistical approaches, the deep learning approach we present here is not parameterised to model spatial autocorrelation specifically. This has benefits, such as freeing us from assumptions of stationarity and isotropy, but the flexibility of our deep neural network could come at a cost in terms of our model's ability to simulate realisations with spatial autocorrelation properties that match those of observations. To investigate this, we have checked the spatial autocorrelation of 250 simulated realisations against that of the held-out test data by comparing variograms (Fig. 7). Each realisation's variogram is calculated by taking its values at the same 10,920 locations as the held-out test observations so as to eliminate any differences that might arise from considering different locations. As is to be expected, each of our realisations displays slightly different spatial autocorrelation properties, which results in a distribution of semi-variances at each lag distance (we are using 2-km bins).

We find that for all lag distances (Fig. 7), the semi-variance of the observations is within the range of the semi-variances of the simulated realisations, suggesting that, overall, there is reasonable agreement between the spatial autocorrelation of realisations and the spatial autocorrelation of the data. To more critical eyes, there is some indication that realisations may on average have slightly too much 'nugget' variance (too much variability at zero distance) while not having quite enough variability at longer ranges (Fig. 7); however, we reserve making more absolute judgements of these higher-order properties of our model's output for further work and testing—in this study, our priority has been point-wise predictive performance and calibration (Fig. 3). For use cases where the spatial autocorrelation of realisations is a priority, we would recommend further investigation into these properties of Bayesian deep learning approaches like ours.

Overall, on the basis of this comparison of simulation and observation variograms using held-out test data (Fig. 7) it appears that, in addition to providing a well-calibrated and sharp predictive distribution (Fig. 3 and Sect. 3), our Bayesian deep learning approach also produces realisations with similar spatial autocorrelation properties to the observations. Each simulated realisation represents values we could observe if we were to repeat the data collection procedure, subject to our current state of knowledge (as represented by the posterior predictive distribution; Gelman et al. 2013).

It is an additional benefit of our Bayesian approach that obtaining realisations comes at no additional computational cost over what is already required to make general predictions with our model. This is because Monte Carlo sampling must be used to obtain our otherwise intractable posterior predictive distribution, and each of these samples is a realisation. So simulation is an innate part of our Bayesian approach. It is also the case that Monte Carlo dropout neural networks are a computationally efficient Bayesian method. On a single GPU workstation, it takes under 30 minutes to train our model on 87,361 training observations. Simulation then takes about 5 seconds per realisation for the entire 0.6 million grid cell map.

References

Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M, Kudlur M, Levenberg J, Monga R, Moore S, Murray DG, Steiner B, Tucker P, Vasudevan V, Warden P, Wicke M, Yu Y, Zheng X (2016) Tensorflow: a system for large-scale machine learning. In: 12th USENIX symposium on operating systems design and implementation (OSDI 16), pp 265–283

Alzubaidi F, Mostaghimi P, Swietojanski P, Clark SR, Armstrong RT (2021) Automated lithology classification from drill core images using convolutional neural networks. J Pet Sci Eng 197:107933

Berger JO (1985) Statistical decision theory and bayesian analysis. Springer, Berlin

British Geological Survey (2020) Geology of Britain Viewer. Accessed through online web interface at http://mapapps.bgs.ac.uk/geologyofbritain/home.html

Cawley GC, Janacek GJ, Haylock MR, Dorling SR (2007) Predictive uncertainty in environmental modelling. Neural Netw 20(4):537–549

Chen Y, Zhu L, Ghamisi P, Jia X, Li G, Tang L (2017) Hyperspectral images classification with gabor filtering and convolutional neural network. IEEE Geosci Remote Sens Lett 14(12):2355–2359

Clark I (2010) Statistics or geostatistics? Sampling error or nugget effect? J South Afr Inst Min Metal 110(6):307–312

Colomina I, Molina P (2014) Unmanned aerial systems for photogrammetry and remote sensing: a review. ISPRS J Photogramm 92:79–97

Cressie N (1990) The origins of kriging. Math Geol 22(3):239–252

de la Varga M, Wellmann JF (2016) Structural geologic modeling as an inference problem: a bayesian perspective. Interpretation 4(3):SM1–SM16

Dillon JV, Langmore I, Tran D, Brevdo E, Vasudevan S, Moore D, Patton B, Alemi A, Hoffman M, Saurous RA (2017) Tensorflow distributions. arXiv preprint arXiv:1711.10604

Dimitrakopoulos R (1998) Conditional simulation algorithms for modelling orebody uncertainty in open pit optimisation. Int J Min Reclam Environ 12(4):173–179

Dimitrakopoulos R (2018) Stochastic mine planning-methods, examples and value in an uncertain world. In: Advances in applied strategic mine planning. Springer, pp 101–115

Fox CR, Ülkümen G (2011) Distinguishing two dimensions of uncertainty. Perspectives on thinking, judging, and decision making 14

Gal Y, Ghahramani Z (2016) Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In: International conference on machine learning, pp 1050–1059

Gelman A, Carlin JB, Stern HS, Dunson DB, Vehtari A, Rubin DB (2013) Bayesian data analysis. CRC Press, Cambridge

Gneiting T, Balabdaoui F, Raftery AE (2007) Probabilistic forecasts, calibration and sharpness. J R Stat Soc B 69(2):243–268

Gneiting T, Katzfuss M (2014) Probabilistic forecasting. Annu Rev Stat Appl 1:125–151

Gneiting T, Raftery AE (2007) Strictly proper scoring rules, prediction, and estimation. J Am Stat Assoc 102(477):359–378

Gotway CA, Hartford AH (1996) Geostatistical methods for incorporating auxiliary information in the prediction of spatial variables. J Agric Biol Environ Stat 1:17–39

Grose L, Ailleres L, Laurent G, Armit R, Jessell M (2019) Inversion of geological knowledge for fold geometry. J Struct Geol 119:1–14

Handcock MS, Stein ML (1993) A bayesian analysis of kriging. Technometrics 35(4):403–410

Hengl T, Heuvelink GB, Rossiter DG (2007) About regression-kriging: from equations to case studies. Comput Geosci 33(10):1301–1315

Hijmans RJ (2017) raster: Geographic Data Analysis and Modeling. R package version 2.6-7

Johnson C, Breward N, Ander E, Ault L (2005) G-base: baseline geochemical mapping of Great Britain and Northern Ireland. Geochem Explor Environ Anal 5(4):347–357

Journel AG (1974) Geostatistics for conditional simulation of ore bodies. Econ Geol 69(5):673–687

Journel AG, Rossi M (1989) When do we need a trend model in kriging? Math Geol 21(7):715–739

Kampffmeyer M, Salberg AB, Jenssen R (2016) Semantic segmentation of small objects and modeling of uncertainty in urban remote sensing images using deep convolutional neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp 1–9

Kendall A, Gal Y (2017) What uncertainties do we need in bayesian deep learning for computer vision? In: Advances in neural information processing systems, pp 5574–5584

Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980

Kirkwood C (2016) A dropout-regularised neural network for mapping arsenic enrichment in SW England using MXNet. NERC open research archive

Kirkwood C (2020) Deep covariate-learning: optimising information extraction from terrain texture for geostatistical modelling applications. arXiv preprint arXiv:2005.11194

Kirkwood C, Cave M, Beamish D, Grebby S, Ferreira A (2016) A machine learning approach to geochemical mapping. J Geochem Explor 167:49–61

Kirkwood C, Economou T, Odbert H, Pugeault N (2021) A framework for probabilistic weather forecast post-processing across models and lead times using machine learning. Philos Trans R Soc A 379(2194):20200099

Krige DG (1951) A statistical approach to some basic mine valuation problems on the witwatersrand. J South Afr Inst Min Metall 52(6):119–139

Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105

Laaha G, Skøien J, Blöschl G (2014) Spatial prediction on river networks: comparison of top-kriging with regional regression. Hydrol Process 28(2):315–324

Lamichhane S, Kumar L, Wilson B (2019) Digital soil mapping algorithms and covariates for soil organic carbon mapping and their implications: a review. Geoderma 352:395–413

LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436–444

LeNail A (2019) Nn-svg: Publication-ready neural network architecture schematics. J. Open Source Softw. 4(33):747

Li T, Shen H, Yuan Q, Zhang X, Zhang L (2017) Estimating ground-level PM2.5 by fusing satellite and station observations: a geo-intelligent deep learning approach. Geophys Res Lett 44(23):11–985

Li Y, Sun Y, Reich BJ (2020) Deepkriging: Spatially dependent deep neural networks for spatial prediction. arXiv preprint arXiv:2007.11972

Luo W, Li Y, Urtasun R, Zemel R (2016) Understanding the effective receptive field in deep convolutional neural networks. In: Advances in neural information processing systems, pp 4898–4906

Matheron G (1962) Traité de géostatistique appliquée. Mémoires du Bureau de Recherches Géologiques et Minières, Éditions Technip

Menabde M, Froyland G, Stone P, Yeates G (2018) Mining schedule optimisation for conditionally simulated orebodies. In: Advances in applied strategic mine planning. Springer, pp 91–100

Mosegaard K, Tarantola A (1995) Monte carlo sampling of solutions to inverse problems. J Geophys Res Solid Earth 100(B7):12431–12447

Mulder V, De Bruin S, Schaepman ME, Mayr T (2011) The use of remote sensing in soil and terrain mapping-a review. Geoderma 162(1–2):1–19

Neal RM (1996) Priors for infinite networks. In: Bayesian learning for neural networks. Springer, pp 29–53

Olierook HK, Scalzo R, Kohn D, Chandra R, Farahbakhsh E, Clark C, Reddy SM, Müller RD (2021) Bayesian geological and geophysical data fusion for the construction and uncertainty quantification of 3d geological models. Geosci Front 12(1):479–493

Padarian J, Minasny B, McBratney AB (2019) Using deep learning for digital soil mapping. Soil 5(1):79–89

Parmentier B, McGill B, Wilson AM, Regetz J, Jetz W, Guralnick RP, Tuanmu MN, Robinson N, Schildhauer M (2014) An assessment of methods and remote-sensing derived covariates for regional predictions of 1 km daily maximum air temperature. Remote Sens 6(9):8639–8670

Perol T, Gharbi M, Denolle M (2018) Convolutional neural network for earthquake detection and location. Sci Adv 4(2):e1700578

Pilz J, Spöck G (2008) Why do we need and how should we implement bayesian kriging methods. Stoch Environ Res Risk Assess 22(5):621–632

Poggio L, Gimona A, Brewer MJ (2013) Regional scale mapping of soil properties and their uncertainty with a large number of satellite-derived covariates. Geoderma 209:1–14

R Core Team (2020) R: a language and environment for statistical computing. R Foundation for Statistical Computing, Vienna

Raftery AE, Lewis SM (1996) Implementing MCMC. Markov chain Monte Carlo in practice, pp 115–130

Ruiz-Arias J, Pozo-Vázquez D, Santos-Alamillos F, Lara-Fanego V, Tovar-Pescador J (2011) A topographic geostatistical approach for mapping monthly mean values of daily global solar radiation: a case study in southern spain. Agric For Meteorol 151(12):1812–1822

Sabins FF (1999) Remote sensing for mineral exploration. Ore Geol Rev 14(3–4):157–183

Sambridge M, Mosegaard K (2002) Monte Carlo methods in geophysical inverse problems. Rev Geophys 40(3):1–29

Schaaf A, de la Varga M, Wellmann F, Bond CE (2021) Constraining stochastic 3-D structural geological models with topology information usingÚpproximate Bayesian computation in GemPy 2.1. Geoscientific Model Dev 14(6):3899–3913

Shamsipour P, Schetselaar E, Bellefleur G, Marcotte D (2014) 3D stochastic inversion of potential field data using structural geologic constraints. J Appl Geophys 111:173–182

Shwartz-Ziv R, Tishby N (2017) Opening the black box of deep neural networks via information. arXiv preprint arXiv:1703.00810

Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958

Stein ML (1999) Interpolation of spatial data: some theory for kriging. Springer, Berlin

Tarantola A, Valette B (1982) Inverse problems = quest for information. J Geophys 50(1):159–170

Van Zyl JJ (2001) The shuttle radar topography mission (SRTM): a breakthrough in remote sensing of topography. Acta Astronaut 48(5–12):559–565

Vann J, Bertoli O, Jackson S (2002) An overview of geostatistical simulation for quantifying risk. In: Proceedings of geostatistical association of Australasia symposium" quantifying risk and error, vol 1, Citeseer, 1

Wadoux AMC (2019) Using deep learning for multivariate mapping of soil with quantified uncertainty. Geoderma 351:59–70

Wadoux AMJ, Padarian J, Minasny B (2019) Multi-source data integration for soil mapping using deep learning. Soil 5(1):107–119

Wellmann JF, De La Varga M, Murdie RE, Gessner K, Jessell M (2018) Uncertainty estimation for a geological model of the sandstone greenstone belt, western Australia-insights from integrated geological and geophysical inversion in a Bayesian inference framework. Geol Soc Spec Publ 453(1):41–56

Wilson AG (2020) The case for Bayesian deep learning. arXiv preprint arXiv:2001.10995

Wu X, Liang L, Shi Y, Fomel S (2019) Faultseg3d: Using synthetic data sets to train an end-to-end convolutional neural network for 3d seismic fault segmentation. Geophysics 84(3):IM35–IM45

Yoe C (2011) Principles of risk analysis: decision making under uncertainty. CRC Press, Cambridge

Young DM, Parry LE, Lee D, Ray S (2018) Spatial models with covariates improve estimates of peat depth in blanket peatlands. PLoS ONE 13(9):e0202691

Youssef AM, Pourghasemi HR, Pourtaghi ZS, Al-Katheeri MM (2016) Landslide susceptibility mapping using random forest, boosted regression tree, classification and regression tree, and general linear models and comparison of their performance at Wadi Tayyah Basin, Asir Region, Saudi Arabia. Landslides 13(5):839–856

Zhang L, Zhang L, Du B (2016) Deep learning for remote sensing data: a technical tutorial on the state of the art. IEEE Trans Geosci Remote Sens 4(2):22–40

Zhu XX, Tuia D, Mou L, Xia GS, Zhang L, Xu F, Fraundorfer F (2017) Deep learning in remote sensing: A comprehensive review and list of resources. IEEE Trans Geosci Remote Sens 5(4):8–36

Zuo R, Xiong Y, Wang J, Carranza EJM (2019) Deep learning and its application in geochemical mapping. Earth Sci Rev 192:1–14