Appendix for 'Environmental Sensor Placement with Convolutional Gaussian Neural Processes'

Tom R. Andersson, Wessel P. Bruinsma, Stratis Markou, James Requeima, Alejandro Coca-Castro, Anna Vaughan, Anna-Louise Ellis, Matthew A. Lazzara, Dani Jones, J. Scott Hosking and Richard E. Turner

A. Data considerations

In this section we provide details on the data sources, preprocessing, and normalisation.

A.1. Data sources

The daily-averaged temperature reanalysis data was obtained from ERA5 (Hersbach et al., 2020). The land mask and elevation field was obtained from the BedMachine dataset (Morlighem, 2020). Antarctic temperature locations from staffed and automatic weather stations were downloaded from `ftp.bas.ac.uk/src/`.

A.2. Data preprocessing

The temperature anomaly data and land/elevation auxiliary data were regridded from lat/lon to a Southern Hemisphere Equal Area Scalable Earth 2 (EASE2) grid at $25\,\mathrm{km}$ resolution and cropping to a size of 280×280. This centres the data on the South Pole.

Temperature anomalies were computed from the absolute temperature values by first computing the daily-average climatology across 1950-2013 (i.e. averaging the absolute temperature over time for each day of year, returning a $366 \times 280 \times 280$ tensor). Then, for each day of year, the climatological average was subtracted from the absolute temperature values, returning anomaly values.

A.3. Data normalisation

To aid the training process, we normalised the data before passing it to the ConvGNP and GP models. The temperature data was normalised from Celsius to a mean of 0 and standard deviation of 1. The elevation field was normalised from metres to values in $[0, 1]$. The land mask already took appropriate normalised values in $\{0, 1\}$. The input coordinates were normalised from metres to take values in $[-1, 1]$.

B. The ConvGNP model

Here we provide details on the ConvGNP training procedure and architecture. A high-level schematic of the ConvGNP forward-pass is shown in Figure 1. We refer the reader to Markou et al. 2022 for further model details.

B.1. Generation of $\mathcal{D}_\tau$ for the training, validation, and test datasets

Each daily-average training dataset $\mathcal{D}_\tau$ was generated by first drawing the integer number of simulated temperature anomaly context points $N_c \sim \mathrm{Unif}\{5, 6, \ldots 500\}$ and target points $N_t \sim \mathrm{Unif}\{3000, 3001, \ldots 4000\}$. Allowing for randomness in N_c encourages the model to learn to deal with both data-sparse and data-rich scenarios. Using a fairly large number of target points ensures there is sufficient signal for learning the covariance structure of the data while not incurring the computational cost of a very large N_t. Next, given the randomly sampled N_c and N_t, the 280×280 ERA5 grid cells were sampled uniformly at random to generate the ERA5 context and target locations, $\boldsymbol{X}_\tau^{(\mathrm{c})}$ and $\boldsymbol{X}_\tau^{(\mathrm{t})}$.

For the training dates, the random seed used for generating $\mathcal{D}_\tau$ is changed every epoch, allowing for an infinitely growing training dataset. In contrast, for the validation and test dates, fixed random seeds are used so that the $\mathcal{D}_\tau$ are deterministically random. This ensures the validation and test metrics are deterministic during and after training.

For the test results given in Table E1, we loop over $N_c \in \{0, 25, 50, \ldots 500\}$ and fix N_t to a value of 2,000. For each setting of N_c, we generate test tasks $\mathcal{D}_\tau$ by looping twice over each day in 2018-2019, resulting in 1,458 test tasks per N_c.

B.2. Antarctic surface temperature anomaly ConvGNP training procedure

The model was trained on data from 1950-2013. An Adam optimiser was used with a learning rate of 8×10^{-5} and an NLL loss function. Gradients with respect to the loss were averaged over batches of two datasets. Validation tasks from 2014-2017 were used for checkpointing the model weights based on the mean NLL over the validation tasks. In total, the ConvGNP was trained for 170 epochs. Training took 11.5 days on a Tesla V100 GPU with a simple implementation of the training pipeline. This could be improved with better computational practices, such as using TensorFlow's graph mode rather than eager mode, which was not supported by the ConvGNP implementation at the time of the experiments.

B.3. ConvGNP architecture

For the ConvGNP model we use the same architecture as described in Markou et al. 2022, except for a few modifications. The U-Net component of the encoder uses 5x5 convolutional kernels with the following sequence of channel numbers (d.s. = 2x2 downsample layer, u.s. = 2x2 upsample layer):

$$128 \xrightarrow{\text{d.s.}} 128 \xrightarrow{\text{d.s.}} 128 \xrightarrow{\text{d.s.}} 128 \xrightarrow{\text{u.s.}} 128 \xrightarrow{\text{u.s.}} 128 \xrightarrow{\text{u.s.}} 128.$$

We use 128 basis functions for the covariance-parameterising neural network, $\boldsymbol{g}$. Using 128 channels for each layer in the U-Net means there are no dimensionality bottlenecks that could reduce the actual rank of the output lowrank covariance matrix. We use bilinear resize operators for the upsampling layers to fix checkerboard artifacts that we encountered when using standard zero-padding upsampling (Odena et al., 2016). For the internal discretisation density of the model, we used 150 points-per-unit (i.e., a 1×1 square of input space contains 150×150 internal discretisation points). We chose 150 points-per-unit to be close to the density of the ERA5 data in normalised coordinates, which is 140×140 in a 1×1 square of input space.[5]

The hyperparameter settings above construct a ConvGNP with 4.16 million learnable parameters. In addition, the choices for the U-Net filter size and internal discretisation density results in a receptive field of over $1500\,\mathrm{km}$. In other words, context observations can influence target predictions in the Gaussian predictive distribution up to roughly $750\,\mathrm{km}$ in either direction of the $\boldsymbol{x}_1$- or $\boldsymbol{x}_2$-dimensions.

B.4. ConvGNP input data

The ConvGNP receives two context sets as input. The first contains observations of the simulated ERA5 daily-average temperature anomaly. The second contains a set of 6 gridded auxiliary and meta-data variables. These are: elevation, land mask, $\cos(\text{day of year})$, $\sin(\text{day of year})$, x_1, and x_2. The $\cos(\text{day of year})$ and $\sin(\text{day of year})$ inputs, where the day of year is normalised between 0 and 2π, together define a circular variable that rotates once per year. This informs the model at what time of year $\mathcal{D}_\tau$ corresponds to, helping with learning seasonal variations in the data. The x_1 and x_2 gridded fields inform the model where in input space the data corresponds to. The gridded auxiliary fields that vary over input space are crucial for allowing the ConvGNP to model spatial non-stationarity. This is because they break the U-Net's translation equivariance property. Future work could explore using *learnable* input auxiliary channels, as in Addison et al. 2022, which could lead to richer non-stationarity at the cost of a greater spatial overfitting risk.

B.5. The SetConv encoder enables fusing data sources with multiple modalities and missing values

The SetConv encoder (Gordon et al., 2020), which fuses the context sets into a single gridded tensor, enables the ConvGNP to model 1) missing data, 2) multiple data streams, and 3) both gridded and off-grid data modalities. This is achieved, in part, through the use of a 'density channel' for each context

[5] Note, finer resolution context data would motivate the use of higher internal discretisation densities.

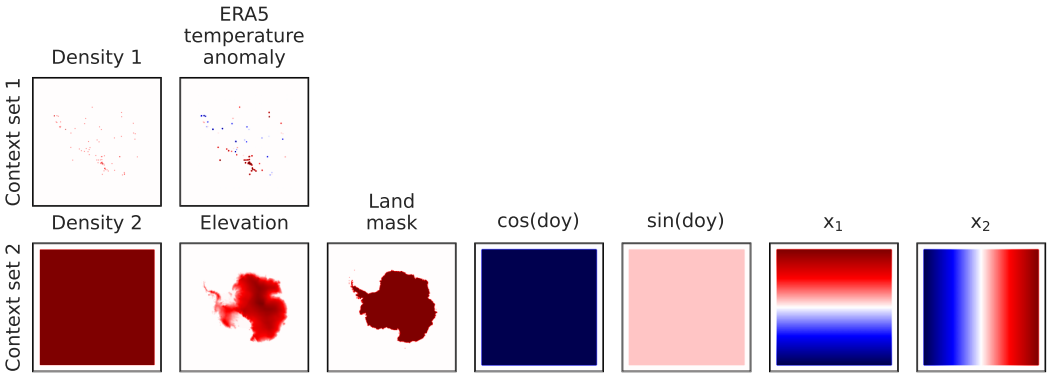

Figure B1. *Example output of the ConvGNP's SetConv encoder. The SetConv was passed a context set with ERA5 temperature anomaly at Antarctic station locations.*

set (Gordon et al., 2020). The density channel measures the density of context points by placing a small Gaussian basis function of unit amplitude at each observation location, such that the density channel takes values close to zero away from observations. Output y-values of context points are encoded in a similar fashion, but the amplitude of the Gaussian basis function is weighted by the value of y. These intermediate functional representations are then discretised onto the model's internal grid to yield the density channels and (N-dimensional) 'data channels' in the gridded encoding. This set-up allows the model to distinguish between the case where an observation is made with a y-value of 0 (data channel is zero but density channel is non-zero), and the case where there is no observation available (both the data and density channels are zero).

Figure B1 shows the channels of a gridded encoding, which are splayed out to highlight the two input context sets provided to the ConvGNP in this study. The density channel for the first context set pinpoints the scattered, point-based temperature anomaly observation locations. The second density channel pertains to the gridded auxiliary context set and thus takes a constant value within the region of data. While in this setting the second gridded context set contained auxiliary variables with no missing data, the SetConv can represent missing data with gridded variables as well as non-gridded variables. For example, missing satellite observations due to cloud cover would be captured by patterns of zeros in the density channel. The density channel can thus be seen as a kind of *missing data channel*, where missing data (e.g. due to sensor malfunction, clouds, or the absence of sensing equipment) is captured with density values close to zero. Therefore, the SetConv encoder equips geospatial deep learning models with an ability to handle missing data, which is an important problem in many application areas (Mitra et al., 2023). However, the degree to which the model can learn to respond to missing data appropriately depends on a training scheme with sufficient examples of missing data, as discussed in Section 4.

C. Gaussian Process benchmarks

Here we provide details on the GP baseline kernels and hyperparameter fitting procedure. All GPs were implemented using the Python package `stheno` (https://github.com/wesselb/stheno) and optimised using the Python package `varz` (https://github.com/wesselb/varz).

C.1. Gibbs GP

The Gibbs kernel (Gibbs, 1997) is a non-stationary generalisation of the EQ kernel. In the $\boldsymbol{x} \in \mathbb{R}^2$ case, the covariance function is given by:

$$k_{\text{Gibbs}}(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 \prod_{i=1}^{2} \left(\frac{2 l_i(\boldsymbol{x}) l_i(\boldsymbol{x}')}{l_i(\boldsymbol{x})^2 + l_i(\boldsymbol{x}')^2} \right)^{1/2} \exp\left(- \sum_{i=1}^{2} \frac{(x_i - x_i')^2}{l_i(\boldsymbol{x})^2 + l_i(\boldsymbol{x}')^2} \right), \tag{C.1}$$

where σ^2 is the variance, and length scale functions $l_1(\boldsymbol{x})$ and $l_2(\boldsymbol{x})$ dictate the length scales in the x_1- and x_2-directions. We parameterise the length scale functions $l_i(\boldsymbol{x})$ as a weighted sum of M regularly placed Gaussian basis functions,

$$l_i(\boldsymbol{x}) = \sum_{m=1}^{M} \theta_{i,m} \exp\left(- \frac{(\boldsymbol{x} - \boldsymbol{x}_m^{(\mu)})^T (\boldsymbol{x} - \boldsymbol{x}_m^{(\mu)})}{2\lambda^2} \right), \tag{C.2}$$

where the $\theta_{i,m}$ are the constrained-positive weights of basis function m for input dimension i, and the basis functions are placed with the $\boldsymbol{x}_m^{(\mu)}$ on a 100×100 grid spanning the input space. The basis function length scale λ is kept fixed and equal to the spacing between basis functions. We note that the basis function spacing controls how quickly the length scale functions can vary, and is a fixed (untrainable) hyperparameter. Too many basis functions can lead to overfitting, while too few can lead to underfitting. We tried different settings and chose a 100×100 grid as the most performant.

We train the parameters $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \sigma\}$, along with the other hyperparameters, using gradient descent on the negative log marginal likelihood (NLML) using an Adam optimiser with learning rate of 5×10^{-3} and a batch size of 10. We used 1950-2013 as a training period, subsampling the dates by a factor of 3, and sampling 500 random context locations for each of the training tasks Appendix B.1. Training was halted after the NLL on validation data spanning 2014–2017 did not improve for 5 epochs.

Figure C1 shows the trained length scale functions $l_1(\boldsymbol{x})$ and $l_2(\boldsymbol{x})$, revealing interesting detail such as very low correlation length scales perpendicular to the coastline.

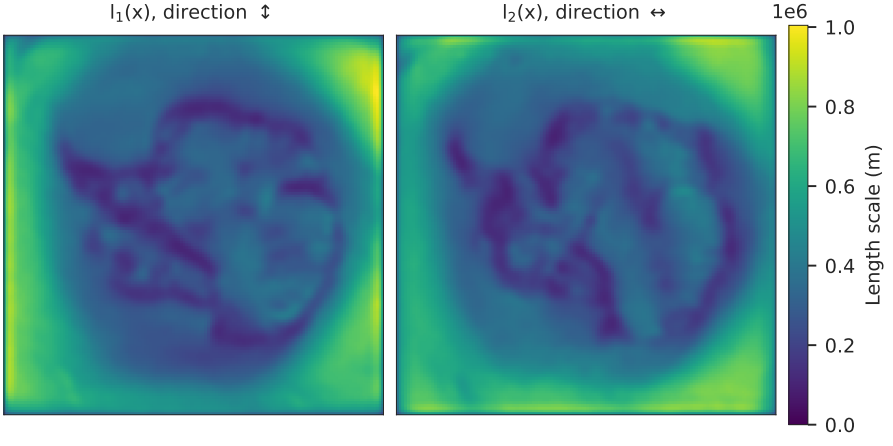

Figure C1. *Learned Gibbs GP length scale functions, $l_1(\boldsymbol{x})$ and $l_2(\boldsymbol{x})$.*

C.2. Exponentiated quadratic and rational quadratic GPs

We also include more simplistic GP baselines using non-isotropic exponentiated quadratic (EQ) and rational quadratic (RQ) kernels, which are both stationary prior covariance functions (unlike the Gibbs kernel). The non-isotropic EQ kernel is:

$$k_{\text{EQ}}(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\frac{(x_1 - x_1')^2}{2\ell_1^2} - \frac{(x_2 - x_2')^2}{2\ell_2^2}\right), \tag{C.3}$$

where σ^2 is the variance, and ℓ_1 and ℓ_2 are the length scales in each input dimension. The non-isotropic RQ kernel is:

$$k_{\text{RQ}}(\mathbf{x}, \mathbf{x}') = \sigma^2 \left(1 + \frac{(x_1 - x_1')^2}{2\alpha\ell_1^2} + \frac{(x_2 - x_2')^2}{2\alpha\ell_2^2}\right)^{-\alpha}, \tag{C.4}$$

where α is the shape parameter, controlling the smoothness of the kernel. The RQ kernel can be seen as an infinite sum of EQ kernels with different length scales (Rasmussen, 2004).

We fit the EQ and RQ GP hyperparameters using the L-BFGS-B algorithm on a batch of 730 dates sampled randomly from 1950-2013. The EQ and RQ GPs are thus exposed to fewer training tasks. However, these models only have a few parameters each. We found that increasing the training set size did not yield improved performance.

D. Non-stationarity in the ConvGNP

The ConvGNP learns richer spatial covariance structure than the GP baselines (Figure D1). Further, while our implementation of the ConvGNP only models correlations over 2D space (i.e. modelling time independently), the model can leverage the day of year auxiliary inputs to learn seasonal non-stationarity in the data (Figure D2). We note that the main changes in the ConvGNP's covariance from summer to winter are caused by changes in the magnitudes of the marginal variances (temperature anomalies take more extreme values in winter). However, the spatial correlations also change (Figure D3). For example, the Ross Ice Shelf site becomes less correlated with the Southern Ocean (Figure D3a), the South Pole becomes more correlated with the surrounding region (Figure D3b), and the East Antarctica site becomes more correlated with the Southern Ocean (Figure D3c).

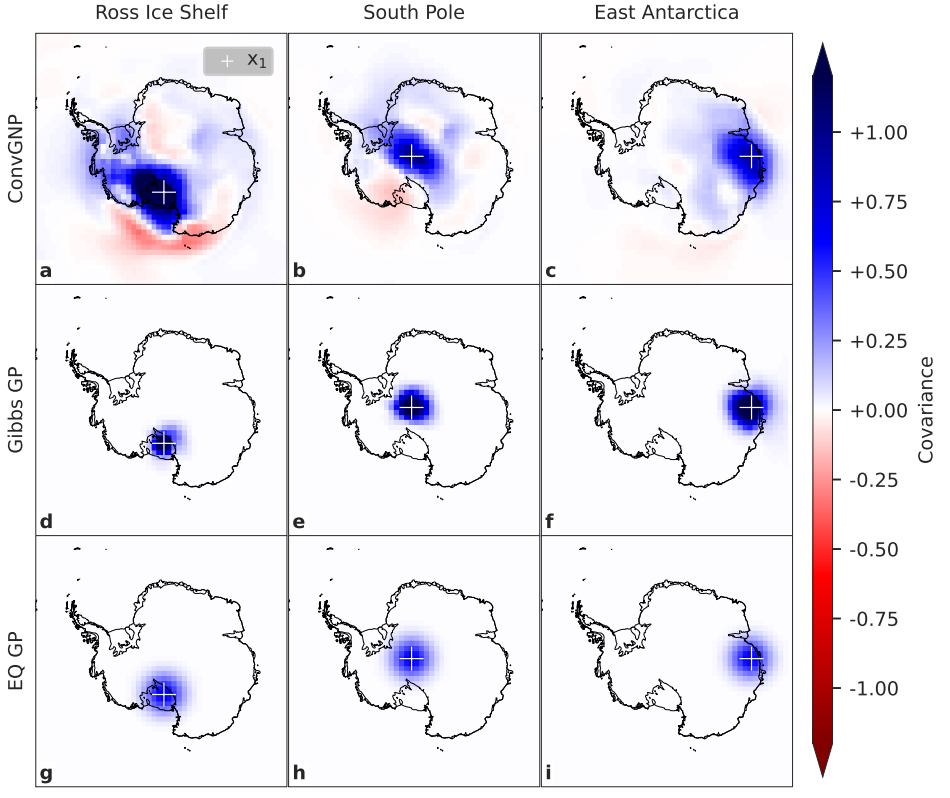

Figure D1. The ConvGNP learns spatially-varying covariance structure. *Heatmaps showing the prior covariance function, $k(\boldsymbol{x}_1, \boldsymbol{x}_2)$, with $\boldsymbol{x}_1$ fixed at the white plus location and $\boldsymbol{x}_2$ varying over the grid. Plots are shown for three different $\boldsymbol{x}_1$-locations (the Ross Ice Shelf, the South Pole, and East Antarctica) and the three models (ConvGNP, Gibbs GP, and EQ GP). The ConvGNP's day of year input was the 1st of June.*

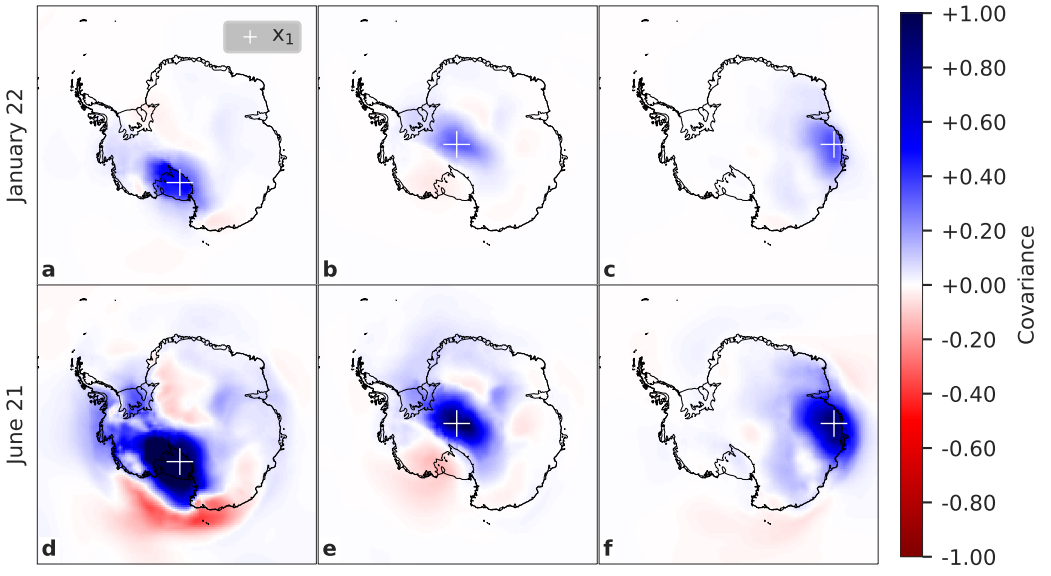

Figure D2. The ConvGNP learns seasonally-varying covariance. *Heatmaps showing the ConvGNP's prior covariance function $k(\boldsymbol{x}_1, \boldsymbol{x}_2)$, as in Figure 3a-c, but for two times of the year: midsummer (Jan 22nd) and midwinter (June 21st). This shows that the ConvGNP has learned a prior covariance function with non-stationarity over day of year.*

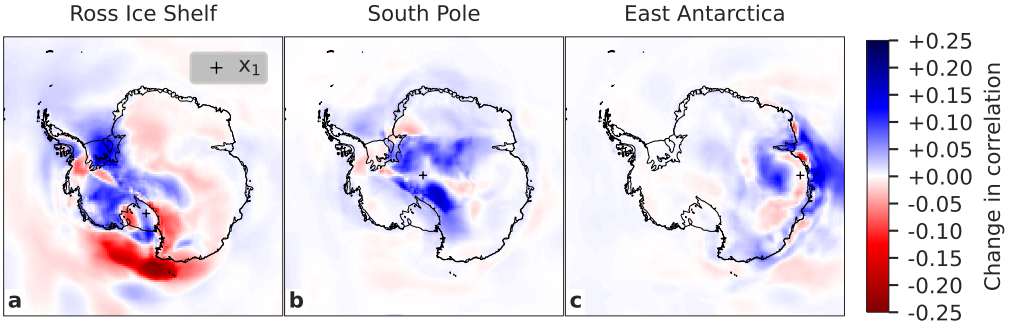

Figure D3. The ConvGNP learns seasonal changes in spatial correlations. *Change in prior correlation from the ConvGNP from the Austral Midsummer (Jan 22nd) to Midwinter (Jun 21st). The correlation ρ was computed from the covariance using $\rho = k(\boldsymbol{x}_1, \boldsymbol{x}_2)/\sqrt{k(\boldsymbol{x}_1, \boldsymbol{x}_1)k(\boldsymbol{x}_2, \boldsymbol{x}_2)}$, with $\boldsymbol{x}_1$ fixed at the black plus location and $\boldsymbol{x}_2$ varying over the grid. Plot shows summer minus winter.*

E. Additional test set results

In this section we provide further details on the models' test set performance.

E.1. Overall performance metrics

Table E1 shows the test set results averaged over $N_c \in \{0, 25, 50, \ldots 500\}$. The ConvGNP outperforms the three GP baselines with statistical significance across all three metrics (the normalised joint NLL, the marginal NLL, and the RMSE).

Table E1. *Performance on test tasks from the period* 2018–2019*, using* $N_c \in \{0, 25, 50, \ldots 500\}$*. Errors indicate standard errors. For each metric, lower is better. Significantly best results in bold.*

METRIC	CONVGNP	GIBBS GP	RQ GP	EQ GP
NORMALISED JOINT NLL	$\mathbf{-0.61}\pm0.00$	-0.42 ± 0.00	-0.23 ± 0.00	0.00 ± 0.00
MARGINAL NLL	$\mathbf{-0.19}\pm0.01$	0.30 ± 0.01	0.47 ± 0.01	0.54 ± 0.01
RMSE (°C)	$\mathbf{1.54}\pm0.01$	1.84 ± 0.01	1.63 ± 0.01	1.72 ± 0.01

E.2. Marginal calibration and sharpness

The calibration and sharpness of probabilistic prediction systems are key performance indicators (Gneiting et al., 2007). To assess these two quantities, we generated test tasks by subsampling the test dates (2018–2019) by a factor of 30 to obtain 25 dates. We then followed the same procedure to generate test tasks as in Appendix B.1–looping over $N_c \in \{0, 25, 50, \ldots 500\}$ with $N_t = 2000$, resulting in 50,000 marginal predictions per N_c for each model.

The calibration of a model's marginal distributions can be assessed using the probability integral transform (PIT). The PIT is defined as the cumulative distribution function (CDF) of the model's marginal distribution evaluated at the true observed value of a particular target point. If a model has perfect calibration, the histogram of PIT values is uniform (Gneiting et al., 2007). When aggregating across all test tasks (with varying N_c values), the ConvGNP's marginal distributions are much better calibrated than the GP baselines, coming closer to the ideal uniform distribution of PIT values (Figure E1).

The sharpness (i.e. degree of certainty) of probabilistic predictions must also be considered alongside calibration; a key goal for probabilistic prediction systems is to maximise sharpness subject to good calibration (Gneiting et al., 2007). We assess marginal distribution sharpness by plotting the standard deviation of the univariate Gaussian marginals against N_c. The ConvGNP makes substantially more confident predictions than the GP baselines across all values of N_c Figure E2. The GP baselines tend to make uninformative predictions with large uncertainty, explaining why their PIT values cluster around 0.5 (Figure E1b–c).

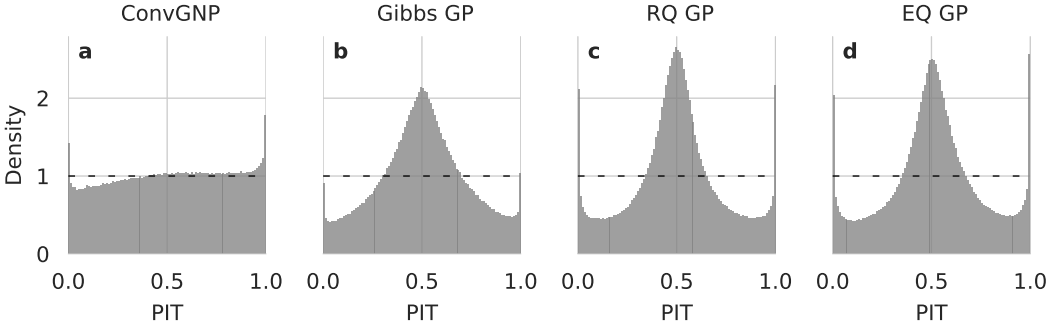

Figure E1. The ConvGNP produces the most well-calibrated marginal predictions. *Probability integral transform (PIT) histograms evaluated on 25 dates from the test years (2018–2019). The PIT is defined as the cumulative distribution function (CDF) of the model's marginal distribution evaluated at the true y-values. A black dashed line shows the ideal uniform distribution (corresponding to perfect calibration).*

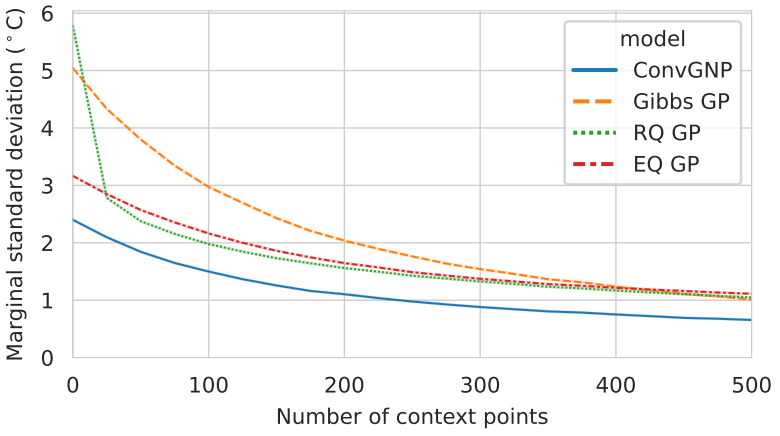

Figure E2. The ConvGNP produces the sharpest marginal predictions. *Mean standard deviation of the models' marginal Gaussian distributions versus number of context points, N_c. 50,000 standard deviation values were used per N_c, derived from 25 dates in the test years (2018–2019). Error bars are standard errors.*

F. Sensor placement acquisition functions

Here we expand upon and mathematically define each acquisition function used for sensor placement.

F.1. Model-based uncertainty reduction acquisition functions

JointMI: Expected reduction in joint entropy over targets after appending the query sensor to the context set C.

$$\alpha(\boldsymbol{x}_i^{(\mathrm{s})}, \tau) = H(\boldsymbol{y}_\tau^{(\mathrm{t})}|C_\tau) - \mathop{\mathbb{E}}_{\pi(y_{\tau,i}^{(\mathrm{s})})}\left[H(\boldsymbol{y}_\tau^{(\mathrm{t})}|C_\tau, \boldsymbol{x}_i^{(\mathrm{s})}, y_{\tau,i}^{(\mathrm{s})})\right] \tag{F.1}$$

$$= H(\boldsymbol{y}_\tau^{(\mathrm{t})}|C_\tau) - \int \pi(y_{\tau,i}^{(\mathrm{s})}; C_\tau) H(\boldsymbol{y}_\tau^{(\mathrm{t})}|C_\tau, \boldsymbol{x}_i^{(\mathrm{s})}, y_{\tau,i}^{(\mathrm{s})})\, \mathrm{d}y_{\tau,i}^{(\mathrm{s})} \tag{F.2}$$

$$\overset{(a)}{\approx} H(\boldsymbol{y}_\tau^{(\mathrm{t})}|C_\tau) - H(\boldsymbol{y}_\tau^{(\mathrm{t})}|C_\tau, \boldsymbol{x}_i^{(\mathrm{s})}, \bar{y}_{\tau,i}^{(\mathrm{s})}) \tag{F.3}$$

$$\overset{(b)}{\approx} c_\tau - H(\boldsymbol{y}_\tau^{(\mathrm{t})}|C_\tau, \boldsymbol{x}_i^{(\mathrm{s})}, \bar{y}_{\tau,i}^{(\mathrm{s})}), \tag{F.4}$$

$$\approx c_\tau - \frac{1}{2}\log((2\pi e)^2|\boldsymbol{K}|), \tag{F.5}$$

$$\approx c_\tau - \frac{1}{2}\log|\boldsymbol{K}|, \tag{F.6}$$

where c_τ is a constant and $\boldsymbol{K}$ is the model's covariance matrix at the target locations after conditioning on the imputed query sensor observation, $(\boldsymbol{x}_i^{(\mathrm{s})}, \bar{y}_{\tau,i}^{(\mathrm{s})})$. In (a) we approximate the intractable expectation integral over the entropy term $H(\boldsymbol{y}_\tau^{(\mathrm{t})}|C_\tau, \boldsymbol{x}_i^{(\mathrm{s})}, y_{\tau,i}^{(\mathrm{s})})$ with a simple substitution of the model's mean prediction $\bar{y}_{\tau,i}^{(\mathrm{s})}$ at query location $\boldsymbol{x}_i^{(\mathrm{s})}$. In (b) we use the fact that $H(\boldsymbol{y}_\tau^{(\mathrm{t})}|C_\tau)$ depends only on τ and not $\boldsymbol{x}_i^{(\mathrm{s})}$. Thus, this placement criterion is equivalent to minimising the entropy over $\boldsymbol{y}_\tau^{(\mathrm{t})}|C_\tau, \boldsymbol{x}_i^{(\mathrm{s})}, \bar{y}_{\tau,i}^{(\mathrm{s})}$ by minimising the determinant of the covariance matrix $\boldsymbol{K}$.

This placement criterion may be hindered by approximating the expectation over the query observation $y_{\tau,i}^{(\mathrm{s})}$ by imputing with the model's mean prediction $\bar{y}_{\tau,i}^{(\mathrm{s})}$ in Equation F.3. Instead, a better scheme

would draw samples from the model's marginal distribution over $y^{(\mathrm{s})}_{\tau,i}$ to estimate the expectation using Monte Carlo sampling, which may better predict the actual reduction in entropy upon conditioning on the true observation. However, Monte Carlo sampling linearly increases the cost of evaluating the acquisition function. Future work should quantify the performance boost from switching to this sampling procedure.

MarginalMI: Expected reduction in marginal entropy over targets upon conditioning on the query sensor. Equivalent to that of Equation F.6, but setting the off-diagonal covariances in the model's output Gaussian distribution to zero when evaluating the entropy term $H(\boldsymbol{y}^{(\mathrm{t})}_{\tau}|C_\tau, \boldsymbol{x}^{(\mathrm{s})}_i, y^{(\mathrm{s})}_{\tau,i})$. Using the resulting independence of the individual marginal distributions after step (b) in Equation A.5 leads to:

$$\alpha(\boldsymbol{x}^{(\mathrm{s})}_i, \tau) \approx c_\tau - \sum_{j=1}^{N_t} H(y^{(\mathrm{t})}_{j,\tau}|C_\tau, \boldsymbol{x}^{(\mathrm{s})}_i, \bar{y}^{(\mathrm{s})}_{\tau,i}), \tag{F.7}$$

$$\approx c_\tau - \sum_{j=1}^{N_t} \frac{1}{2}\log(2\pi e \sigma^2_{\tau,j}), \tag{F.8}$$

$$\approx c_\tau - \sum_{j=1}^{N_t} \log(\sigma^2_{\tau,j}), \tag{F.9}$$

where $\sigma^2_{\tau,j}$ is the variance of the model's marginal Gaussian distribution of target point j at time τ after conditioning on the imputed query sensor observation, $(\boldsymbol{x}^{(\mathrm{s})}_i, \bar{y}^{(\mathrm{s})}_{\tau,i})$. In other words, the acquisition function is the negative sum of the marginal Gaussian entropies at each target location after adding the query sensor to the context set. After this acquisition function is averaged over time steps in Equation 2, the placement criterion amounts to minimising the mean log-variance over time and target locations.

DeltaVar: Expected reduction in mean marginal variance over targets upon conditioning on the query sensor. Following the same expectation approximation as `JointMI` and `MarginalMI` we arrive at:

$$\alpha(\boldsymbol{x}^{(\mathrm{s})}_i, \tau) \approx c_\tau - \frac{1}{N_t}\sum_{j=1}^{N_t} \sigma^2_{\tau,j}. \tag{F.10}$$

The main difference with the `MarginalMI` acquisition function is that `DeltaVar` minimises the absolute marginal variances rather than the log marginal variances.

F.1.1. Model-based uncertainty reduction acquisition functions from the sensor placement experiment

Figure F1 plots heatmaps of the three model-based acquisition functions (`JointMI`, `MarginalMI`, and `DeltaVar`) at the first greedy iteration for the ConvGNP, Gibbs GP, and EQ GP. Also shown are the $K = 10$ proposed sensor placements with the criterion of maximising these acquisition functions.

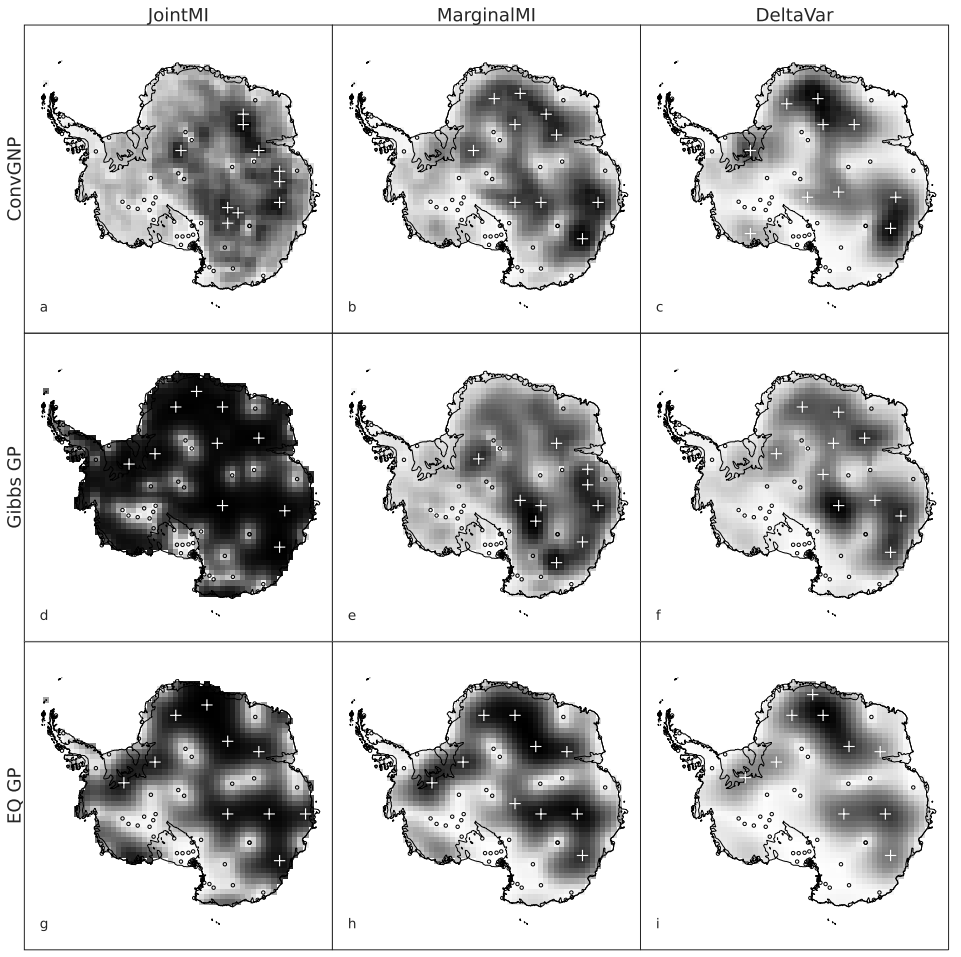

Figure F1. Acquisition functions and sensor placements for all three models. *Maps of acquisition function values $\alpha(\boldsymbol{x}_i^{(s)})$ for the initial $k = 1$ greedy iteration. The initial context set $\boldsymbol{X}^{(c)}$ is derived from real Antarctic station locations (indicated by black crosses). Running the sensor placement algorithm for $K = 10$ sensor placements results in the proposed sensor placements $\boldsymbol{X}^*$ (indicated by pluses).*

F.2. Baseline acquisition functions

ContextDist: Euclidean distance from the closest context point,

$$\alpha(\boldsymbol{x}_i^{(\mathrm{s})}, \tau) = \min\{||\boldsymbol{x}_i^{(\mathrm{s})} - \boldsymbol{x}_{\tau,1}^{(\mathrm{c})}||_2, ..., ||\boldsymbol{x}_i^{(\mathrm{s})} - \boldsymbol{x}_{\tau,N_c}^{(\mathrm{c})}||_2\}, \tag{F.11}$$

Random: Uniform at random in [0, 1],

$$\alpha(\boldsymbol{x}_i^{(\mathrm{s})}, \tau) = u_{\tau,i} \quad \text{where} \quad u_{\tau,i} \sim \mathrm{Unif}(0, 1). \tag{F.12}$$

Maximising this random acquisition function results in placements that are sampled uniformly from the search points $\boldsymbol{X}^{(\mathrm{s})}$.

F.3. Oracle acquisition functions

Let a performance metric γ take in the probability distribution over targets output by prediction map π and the true target values $\boldsymbol{y}_\tau^{(\mathrm{t})}$. Considering only the 1D context set corresponding to observations of the

target variable for notational simplicity, the oracle acquisition functions are:

$$\alpha_{\text{oracle}}(\boldsymbol{x}_i^{(\text{s})}, \tau) = \gamma(\pi(\boldsymbol{y}_\tau^{(\text{t})}; \boldsymbol{X}_\tau^{(\text{c})}, \boldsymbol{y}_\tau^{(\text{c})}, \boldsymbol{X}_\tau^{(\text{t})}), \boldsymbol{y}_\tau^{(\text{t})}) - \gamma(\pi(\boldsymbol{y}_\tau^{(\text{t})}; \{\boldsymbol{X}_\tau^{(\text{c})}, \boldsymbol{x}_i^{(\text{s})}\}, \{\boldsymbol{y}_\tau^{(\text{c})}, y_{\tau,i}^{(\text{s})}\}, \boldsymbol{X}_\tau^{(\text{t})}), \boldsymbol{y}_\tau^{(\text{t})}), \tag{F.13}$$

$$= c_\tau - \gamma(\pi(\boldsymbol{y}_\tau^{(\text{t})}; \{\boldsymbol{X}_\tau^{(\text{c})}, \boldsymbol{x}_i^{(\text{s})}\}, \{\boldsymbol{y}_\tau^{(\text{c})}, y_{\tau,i}^{(\text{s})}\}, \boldsymbol{X}_\tau^{(\text{t})}), \boldsymbol{y}_\tau^{(\text{t})}), \tag{F.14}$$

which is the decrease in performance metric (assuming lower is better) induced by concatenating the true observation $(\boldsymbol{x}_i^{(\text{s})}, y_{\tau,i}^{(\text{s})})$ to the context set. $\alpha_{\text{oracle}}(\boldsymbol{x}_i^{(\text{s})}, \tau)$ is then averaged over τ as in Equation (2) to obtain $\alpha_{\text{oracle}}(\boldsymbol{x}_i^{(\text{s})})$.

F.4. Comment on the dependence on context observations in the acquisition functions

The posterior covariance function of a vanilla GP depends only on the input locations $\boldsymbol{X}^{(\text{c})}$ of the context set, not the observed values $\boldsymbol{y}^{(\text{c})}$. Consequently, the `JointMI`, `MarginalMI`, and `DeltaVar` placement methods will depend only on the input locations. This behaviour is noted by MacKay 1992 for a Bayesian linear regression model with a Gaussian prior, which is a special case of a GP (Rasmussen, 2004). This could be seen as an inflexible limitation of GPs; they cannot augment their posterior correlation structure based on the y-values observed at the $\boldsymbol{x}$-locations. For example, if an extreme y-value is observed in the context set, a GP posterior cannot become more uncertain, which may be a desirable characteristic. By construction, the ConvGNP is a non-linear map from context sets to GPs, which means that the whole GP, including the covariance, can depend on every aspect of the context set, including the y-values. The ConvGNP's y-dependence necessitates the expectation integral over the unobserved query y-value in Equation F.1, as well as the averaging over multiple time steps for the uncertainty-based acquisition functions in Equation 2. Neither of these steps are necessary for the GP baselines since their covariance is independent of the y-values.

G. Oracle sensor placement results

Here we provide more detailed plots from the oracle acquisition function experiment described in Section 3.2.1. Heatmaps of the temporally-averaged $\alpha(\boldsymbol{x}_i^{(s)})$ acquisition functions for the non-oracle and oracle acquisition functions for the ConvGNP, Gibbs GP, and EQ GP are shown in Figure G2, Figure G3, and Figure G4, respectively. Figure G5, Figure G6, and Figure G7 show scatter plots of non-oracle acquisition functions against `OracleJointNLL`, `OracleMarginalNLL`, and `OracleRMSE`, respectively.

We repeat the Pearson correlation analysis of Figure 6 but using a correlation coefficient specifically suited to rankings, the Kendall rank correlation coefficient:

$$\kappa = \frac{1}{N_{\text{pairs}}} \sum_{i<j} \text{sgn}(\alpha_i - \alpha_j)\text{sgn}(\alpha_{\text{oracle},i} - \alpha_{\text{oracle},j}), \tag{G.1}$$

where $N_{\text{pairs}} = S(S-1)/2$ is the total number of pairs, $\alpha_i = \alpha(x_i^{(s)})$, and sgn is the sign function which is $+1$ if the argument is positive and -1 if the argument is negative.. This loops over all pairs of $(\alpha_i, \alpha_{\text{oracle},i})$ and $(\alpha_j, \alpha_{\text{oracle},j})$, checking whether the α values are ranked in the same order as the α_{oracle} values. If so, the pair is 'concordant' and contributes a $+1$ to the sum in Equation (G.1). Otherwise, it is 'discordant' and contributes a -1. Defining the total number of concordant pairs as N_{con}, Equation G.1 can be rewritten as:

$$\kappa = \frac{1}{N_{\text{pairs}}}\big(N_{\text{con}} - (N_{\text{pairs}} - N_{\text{con}})\big), \tag{G.2}$$

$$= 2 \times \frac{N_{\text{con}}}{N_{\text{pairs}}} - 1, \tag{G.3}$$

which we see as the fraction of pairs that are concordant, $N_{\text{con}}/N_{\text{pairs}}$, normalised to lie in $(-1, 1)$.

Identical rankings yield $\kappa = 1$, exactly opposite rankings yield $\kappa = -1$, and if the two rankings are independent the expected value of κ is zero. We computed κ in Python using the `scipy.stats.kendalltau` function. The results are shown in Figure G1.

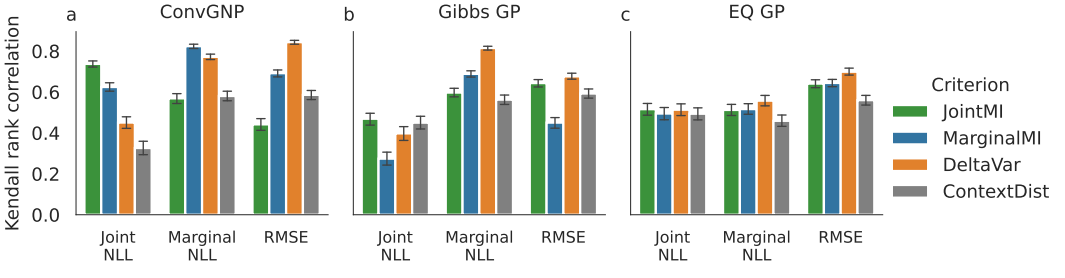

Figure G1. The ConvGNP can reliably rank the value of new observations. *Kendall rank correlation coefficient κ between model-based and oracle acquisition functions. Error bars indicate the 95% percentile interval over 5000 bootstrapped correlation values by resampling the 1365 pairs of points with replacement, measuring how spatially consistent κ is across space.*

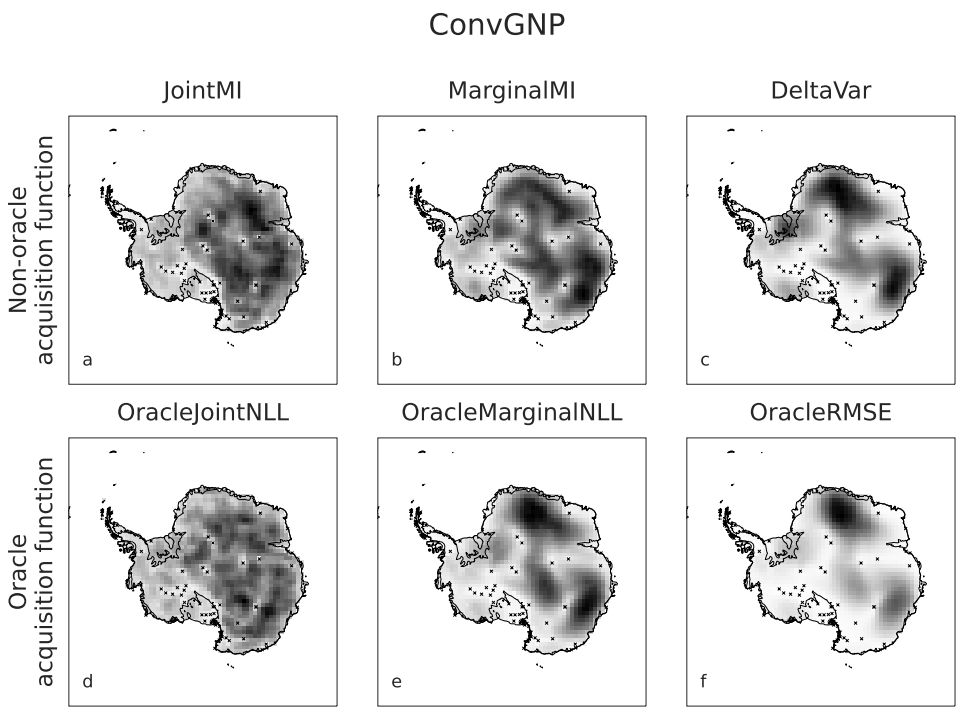

Figure G2. *Non-oracle and oracle acquisition functions for the ConvGNP.*

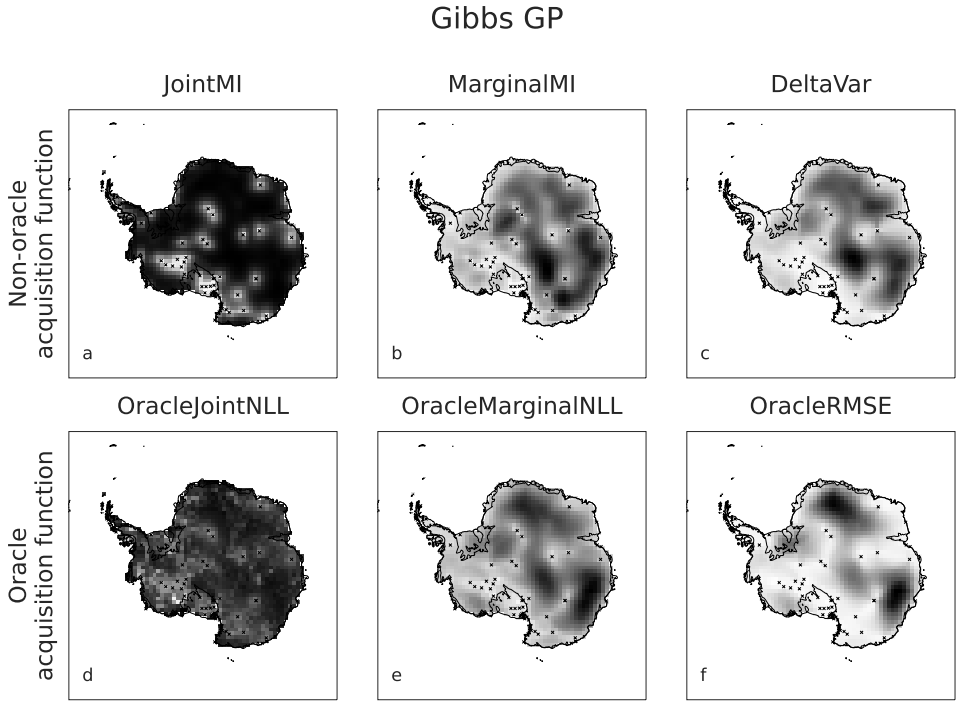

Figure G3. *Non-oracle and oracle acquisition functions for the Gibbs GP.*

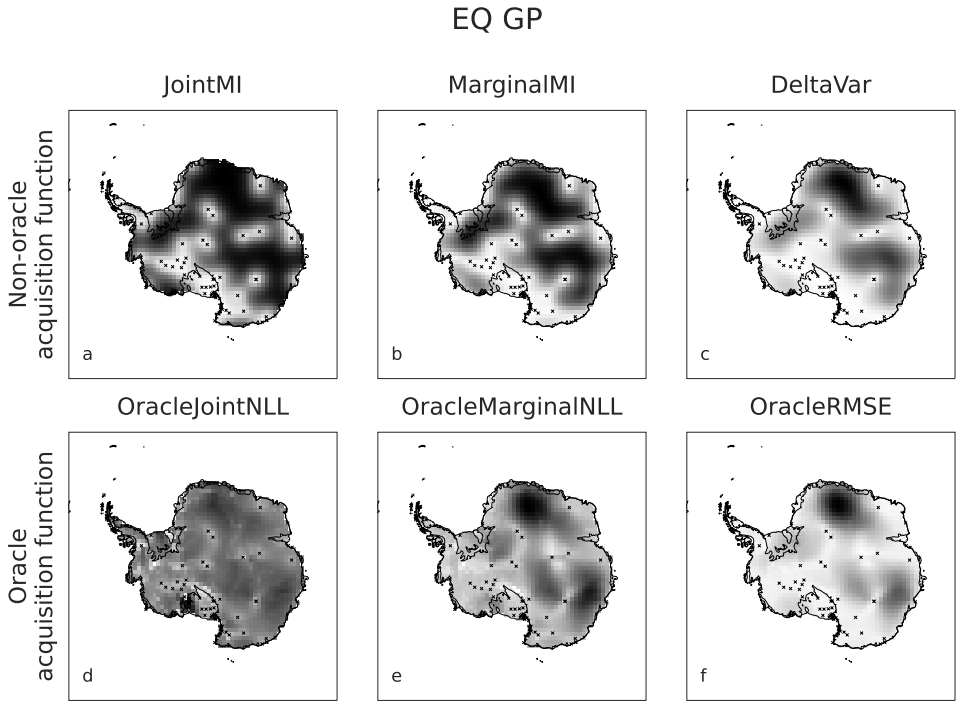

Figure G4. *Non-oracle and oracle acquisition functions for the EQ GP.*

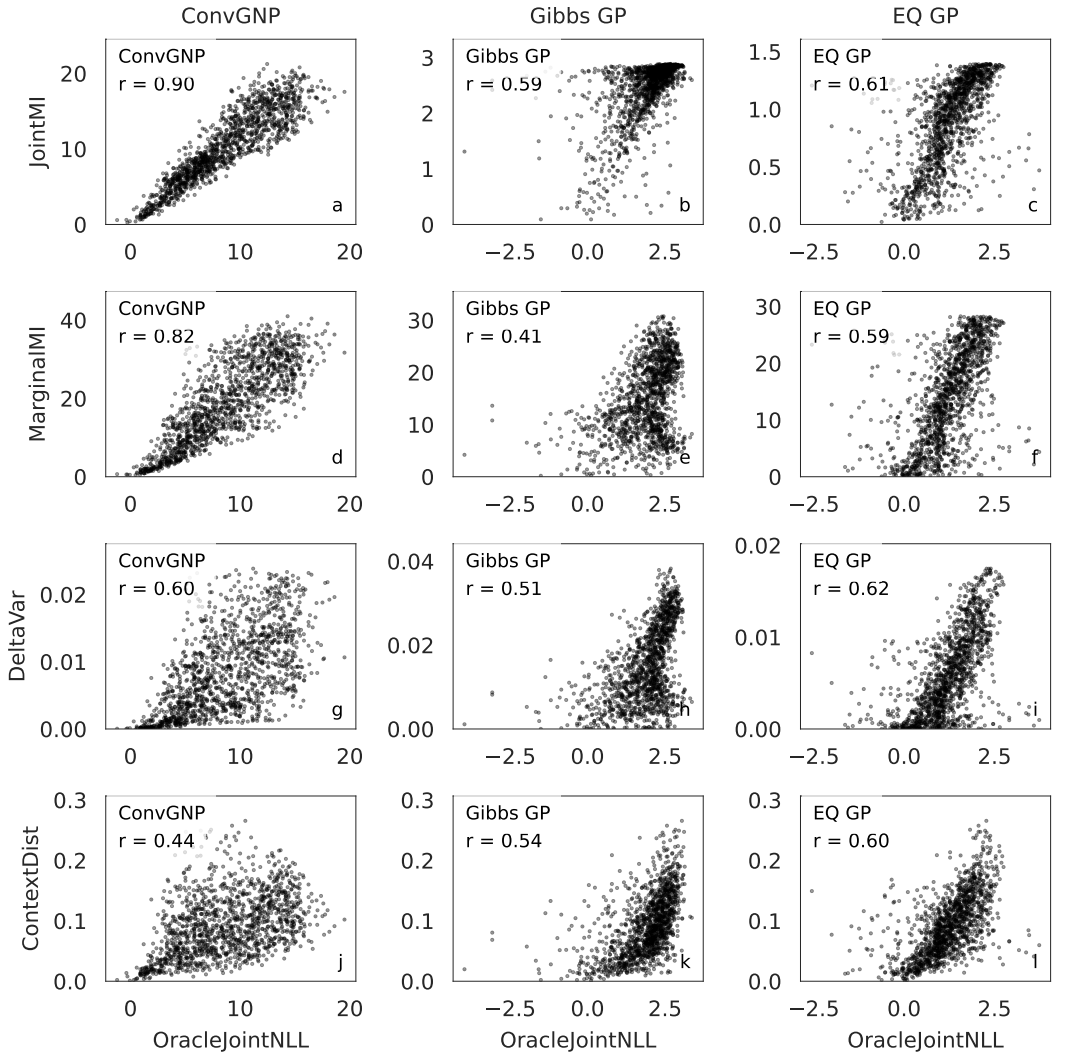

Figure G5. *Scatter plots and correlations between non-oracle acquisition functions and* `OracleJointNLL`.

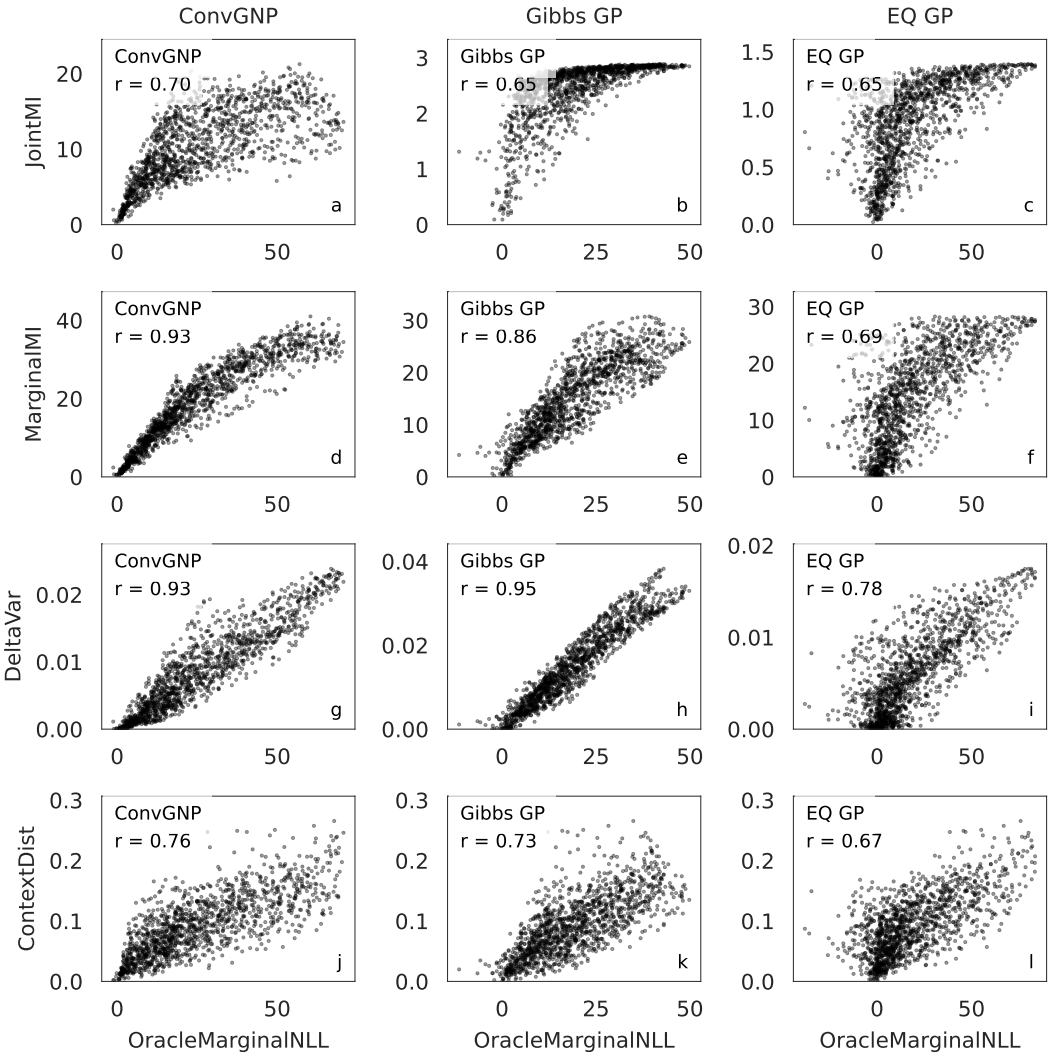

Figure G6. *Scatter plots and correlations between non-oracle acquisition functions and* `OracleMarginalNLL`.

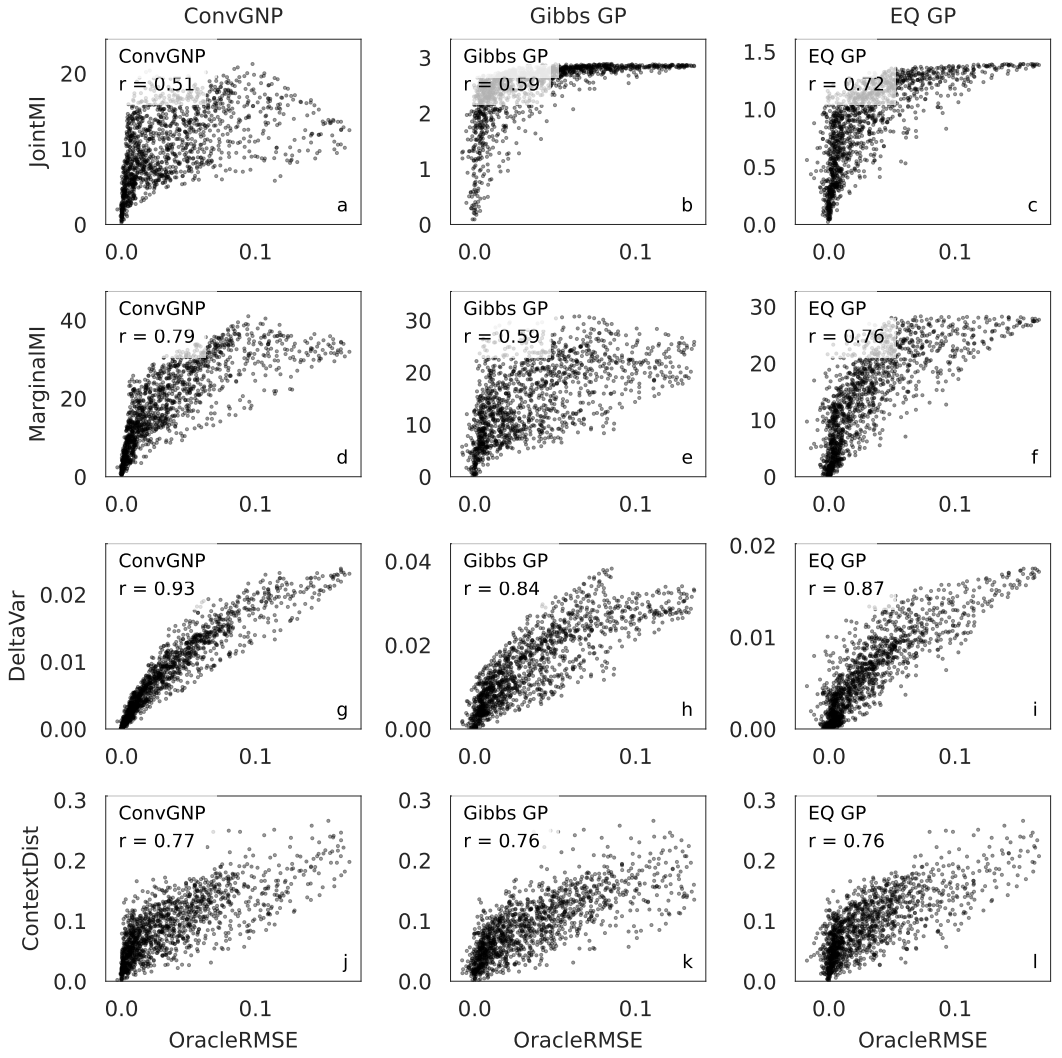

Figure G7. *Scatter plots and correlations between non-oracle acquisition functions and* `OracleRMSE`.

H. Sensor placement toy experiment details

Here we provide more details on the sensor placement toy experiment.

H.1. Experiment design choices

To emulate a non-uniform, real-world sensor network to be optimised, we initialise $\boldsymbol{X}_\tau^{(c)}$ at the locations of real Antarctic temperature station observations at $\tau = 2009/02/15$, and interpolate the gridded ERA5 temperature anomaly at $\boldsymbol{X}_\tau^{(c)}$ to compute $\boldsymbol{y}_\tau^{(c)}$.

The `JointMI`, `MarginalMI`, and `DeltaVar` criteria for the GP models and the `ContextDist` criterion only depend on the context set locations, $\boldsymbol{X}_\tau^{(c)}$, not the observed values $\boldsymbol{y}_\tau^{(t)}$ (Appendix F.4). Since we have a non-varying $\boldsymbol{X}_\tau^{(c)}$ in this toy experiment, the model-based acquisition functions do not depend on time τ for the GPs, and so the sensor placements for these criteria can be run using a single task. In contrast, each oracle acquisition function and the ConvGNP's model-based criteria *do* depend on the observed values $\boldsymbol{y}_\tau^{(t)}$. For these acquisition functions we compute the average $\alpha(\boldsymbol{x}_i^{(s)})$ values in Equation 2 using dates in 2014–2017, subsampled by a factor of 14, to yield $J = 105$ sensor placement search tasks. A regular spatial grid is used for the search space $\boldsymbol{X}^{(s)}$, with one query location every $100\,\mathrm{km}$. The $\boldsymbol{x}_i^{(s)}$ were masked out over the ocean to focus on land stations. The target locations $\boldsymbol{X}_\tau^{(t)}$ were defined on the same grid with points over ocean masked out to focus on predicting land surface temperature. These choices result in a search size and target set size of $S = N_t = 1,365$. Note, limiting the target set size to $N_t = 1,365$ was due to the cubic computational cost of the GP baselines–the ConvGNP could use a much denser target grid due to its linear scaling with number of target points. For the ConvGNP, sequentially computing one of the acquisition functions over these $J = 105$ dates and $S = 1,365$ search points (totalling $143,325$ forward passes) took roughly 3 hours on a 32 GB NVIDIA V100 GPU using TensorFlow's eager mode.

The proposed placements $\boldsymbol{X}^*$ were assessed by analysing model performance over 243 uniformly spaced dates in 2018–2019 (sampling every 3rd day). The sensor placement search period aligns with the model validation period, while the sensor placement analysis period aligns with the model test period.

H.2. Full sensor placement results

Figure F1 plots the $K = 10$ proposed sensor placements for each model and placement criterion. The full breakdown of the sensor placement results for each model, metric, and criterion is shown in Figure H1, using independent y-axes to highlight differences in placement criterion performance for a given model. However, this visually obscures two other differences: initial model performance and the scale of improvement with added stations. Plotting the results with the y-axes shared across models highlights these differences (Figure H2).

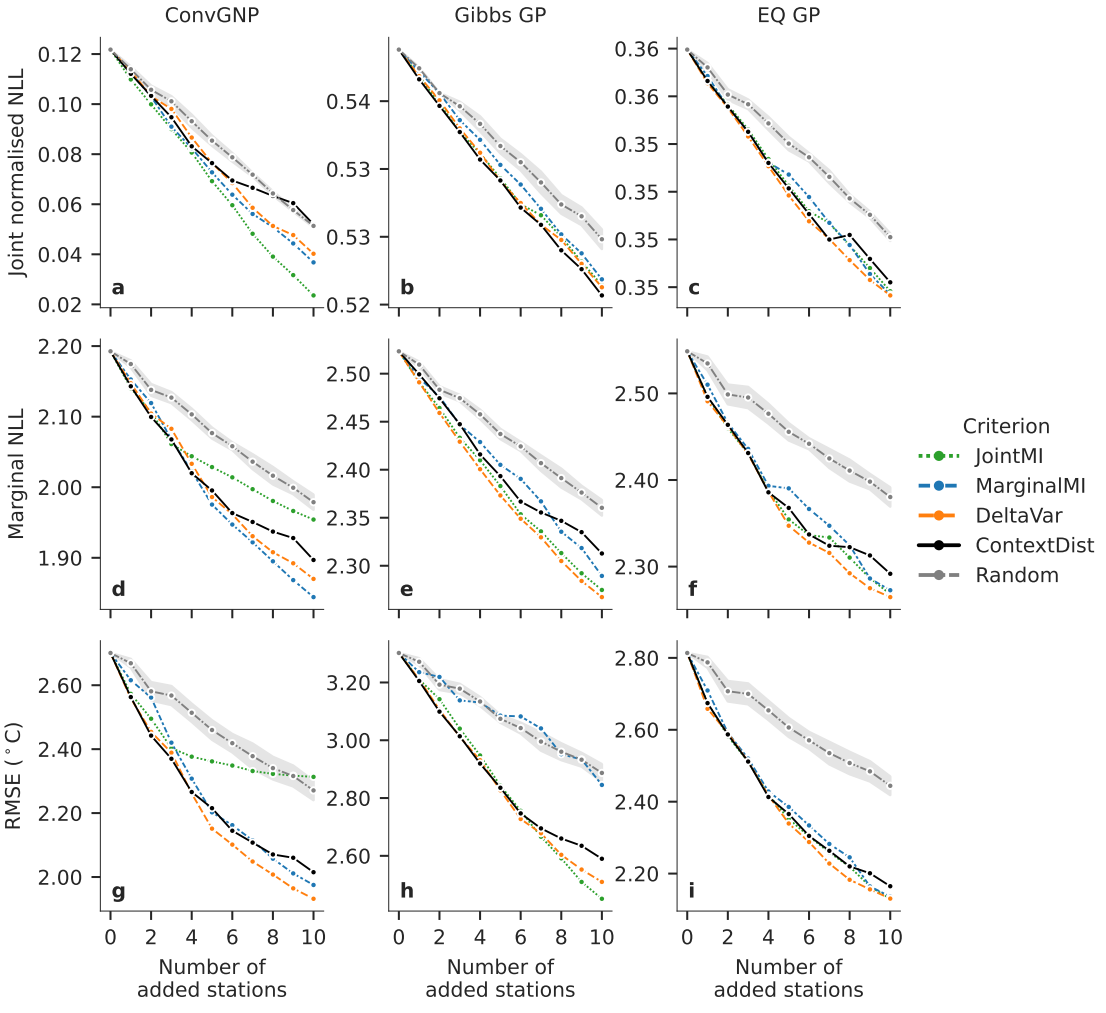

Figure H1. Sensor placement results. *Performance metrics on the 2018-2019 sensor placement ERA5 test data versus the number of stations revealed to the models. Results are averaged over tasks with targets defined on a regular grid over Antarctica. For the* `Random` *placement criterion, the confidence interval shows the standard error based on 5 random placements.* ***a-c****, joint normalised negative log-likelihood (NLL).* ***d-f****, mean marginal NLL.* ***g-i****, root mean squared error (RMSE).*

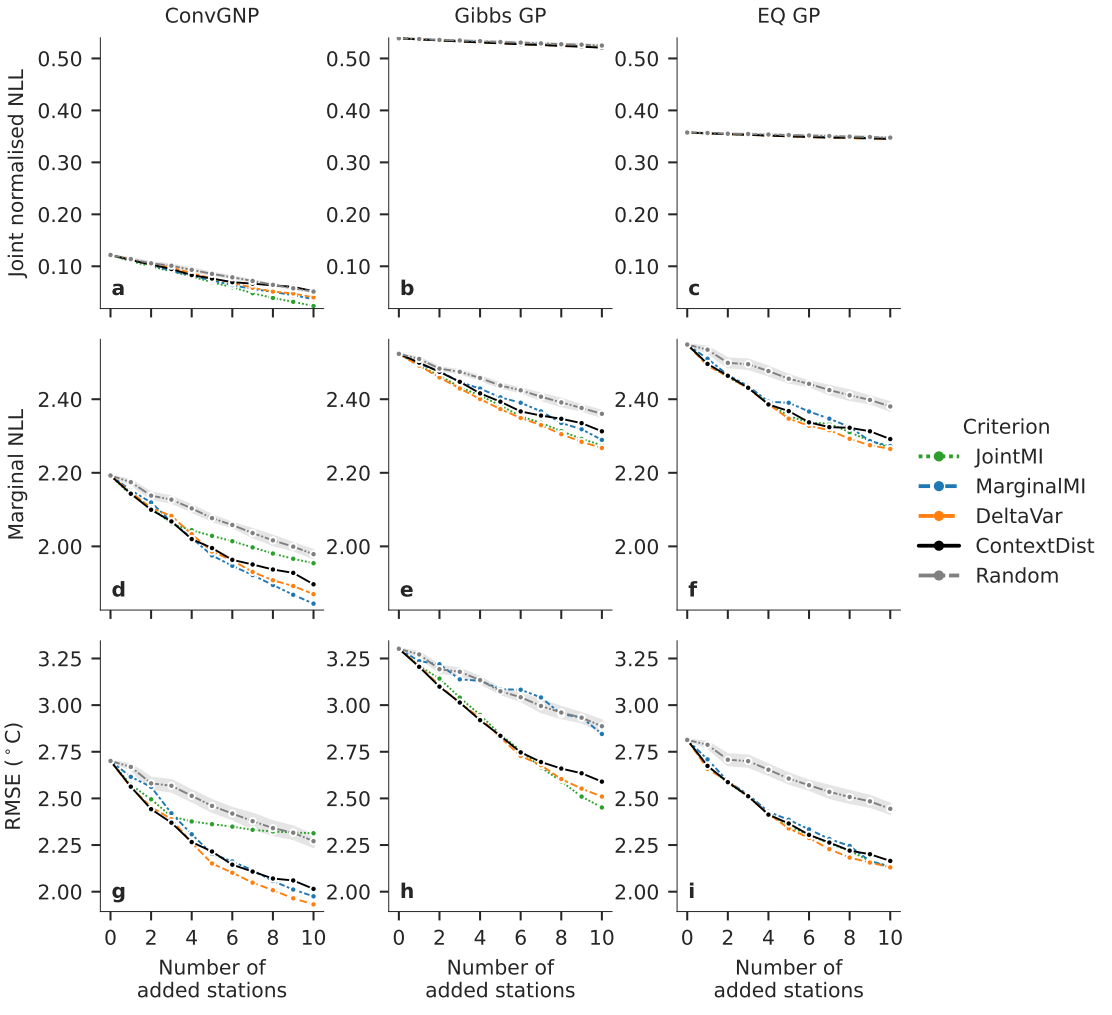

Figure H2. Sensor placement results with shared y-axes. *Performance metrics on the 2018-2019 sensor placement ERA5 test data versus the number of stations revealed to the models. Placements are revealed in the order of placement for each criterion. Results are averaged over tasks. For the* `Random` *placement criterion, the confidence interval shows the standard error based on 5 random placements.* ***a-c****, joint normalised negative log-likelihood (NLL).* ***d-f****, mean marginal NLL.* ***g-i****, root mean squared error (RMSE).*

I. Comparison of the ConvGNP with deep kernel learning

The ConvGNP is perhaps most comparable to deep kernel learning (DKL; Wilson et al. 2015). In DKL, neural networks parameterise a non-stationary covariance function and are trained to optimise a GP prior over the context data. Then, as with vanilla GPs, standard Bayes' rule conditioning is used with that prior to output posterior GP predictives. In contrast, the ConvGNP learns to directly output the GP predictive during training. This allows for outputting GPs that are not in the class of conditioned GP priors, which is much more flexible and aids modelling complex environmental data. However, since robust conditioning is not built in to the ConvGNP, it must learn appropriate conditioning mechanics from the data. This necessitates a novel training scheme where the model is provided with a range of context scenarios, expanding the training design space and likely making the ConvGNP more data-hungry than DKL.

The ConvGNP scales linearly with the number of context points due to the SetConv encoder and neural network architecture used to output the GP predictive. Predictions with the ConvGNP's GP predictive are made scalable by directly learning to output a low-rank approximation of the covariance, reducing the computational cost from cubic to linear. In contrast, DKL methods are by default cubic in the number of context and target points and must use approximate inference on the exact GP to make predictions scalable (Wilson et al., 2015), resulting in an unknown penalty to prediction quality (Wang et al., 2019). It is not obvious which is the best approach, although the out-of-the-box nature of the ConvGNP's scalability is convenient from a practitioner's point of view. Computational cost at inference time is important in the context of environmental applications because observations and target predictions locations may lie on dense grids.

DKL can also be deployed in a meta-learning fashion (Patacchiola et al., 2020), which mitigates the risk of overfitting that comes with heavily-parameterised covariance functions (Ober et al., 2021). A direct comparison between the meta-learning abilities of the ConvGNP and DKL has not been performed and would be a valuable addition to the literature.

Appendix References

Addison, H., Kendon, E., Ravuri, S., Aitchison, L., and Watson, P. A. (2022). Machine learning emulation of a local-scale UK climate model. In *Tackling Climate Change with Machine Learning workshop at NeurIPS 2022*. arXiv. arXiv:2211.16116 [physics]

Gibbs, M. (1997). Bayesian gaussian processes for regression and classication. *PhD Thesis*

Gneiting, T., Balabdaoui, F., and Raftery, A. E. (2007). Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(2):243–268

Gordon, J., Bruinsma, W. P., Foong, A. Y. K., Requeima, J., Dubois, Y., and Turner, R. E. (2020). Convolutional Conditional Neural Processes. In *International Conference on Learning Representations*

Hersbach, H., Bell, B., Berrisford, P., Hirahara, S., Horányi, A., Muñoz-Sabater, J., Nicolas, J., Peubey, C., Radu, R., Schepers, D., Simmons, A., Soci, C., Abdalla, S., Abellan, X., Balsamo, G., Bechtold, P., Biavati, G., Bidlot, J., Bonavita, M., De Chiara, G., Dahlgren, P., Dee, D., Diamantakis, M., Dragani, R., Flemming, J., Forbes, R., Fuentes, M., Geer, A., Haimberger, L., Healy, S., Hogan, R. J., Hólm, E., Janisková, M., Keeley, S., Laloyaux, P., Lopez, P., Lupu, C., Radnoti, G., de Rosnay, P., Rozum, I., Vamborg, F., Villaume, S., and Thépaut, J.-N. (2020). The ERA5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146(730):1999–2049

MacKay, D. J. C. (1992). Information-based objective functions for active data selection. *Neural Computation*, 4(4):590–604

Markou, S., Requeima, J., Bruinsma, W. P., Vaughan, A., and Turner, R. E. (2022). Practical Conditional Neural Processes Via Tractable Dependent Predictions. In *The Tenth International Conference on Learning Representations*

Mitra, R., McGough, S. F., Chakraborti, T., Holmes, C., Copping, R., Hagenbuch, N., Biedermann, S., Noonan, J., Lehmann, B., Shenvi, A., Doan, X. V., Leslie, D., Bianconi, G., Sanchez-Garcia, R., Davies, A., Mackintosh, M., Andrinopoulou, E.-R., Basiri, A., Harbron, C., and MacArthur, B. D. (2023). Learning from data with structured missingness. *Nature Machine Intelligence*, 5(1):13–23

Morlighem, M. (2020). MEaSUREs BedMachine Antarctica, Version 2. *NASA National Snow and Ice Data Center DAAC*

Ober, S. W., Rasmussen, C. E., and Wilk, M. v. d. (2021). The promises and pitfalls of deep kernel learning. In *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, pages 1206–1216. PMLR. ISSN: 2640-3498

Odena, A., Dumoulin, V., and Olah, C. (2016). Deconvolution and Checkerboard Artifacts. *Distill*, 1(10):e3

Patacchiola, M., Turner, J., Crowley, E. J., O'Boyle, M., and Storkey, A. (2020). Bayesian Meta-Learning for the Few-Shot Setting via Deep Kernels. In *34th Conference on Neural Information Processing Systems (NeurIPS 2020)*. arXiv:1910.05199 [cs, stat]

Rasmussen, C. E. (2004). Gaussian Processes in Machine Learning. Springer

Wang, K., Pleiss, G., Gardner, J., Tyree, S., Weinberger, K. Q., and Wilson, A. G. (2019). Exact Gaussian Processes on a Million Data Points. In *Advances in Neural Information Processing Systems*, volume 32

Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. (2015). Deep Kernel Learning. In *Artificial Intelligence and Statistics (AISTATS)*. arXiv:1511.02222 [cs, stat]