Supplementary Materials: Various-Neighbor Neural Network Emulator via Feature Engineering and Data Augmentation for Computer Models with High Dimensional Outputs

S1 ADAM Optimizer Algorithm

In this section, we provide a summary of the ADAM optimizer algorithm used to estimate $\boldsymbol{\omega}$.

Algorithm 1 The updating rule of ADAM

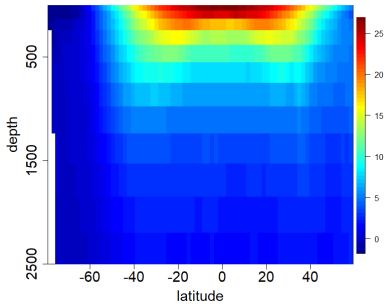
Require: α: Stepsize
Require: $0 \leq \beta_1, \beta_2 < 1$: Exponential decay rates
Require: $\boldsymbol{\ell}_{\mathrm{r}}(\boldsymbol{\omega})$: Objective function with parameters $\boldsymbol{\omega}$
Require: Initialization;
$\boldsymbol{\omega}_0$: Initial parameter vector
$\mathbf{M}_0 \leftarrow 0$: Initialize the first moment vector
$\mathbf{V}_0 \leftarrow 0$: Initialize the second moment vector
$t \leftarrow 0$: Initialize time-step
while $\boldsymbol{\omega}_t$ not converged **do**
$t \leftarrow t + 1$
$\mathbf{g}_t \leftarrow \nabla_{\boldsymbol{\omega}} \boldsymbol{\ell}_{\mathrm{r}}(\boldsymbol{\omega_{t-1}})$: computing gradients of objective function at time-step t
$\mathbf{M}_t \leftarrow \beta_1 \mathbf{M}_{t-1} + (1 - \beta_1) g_t$: Update the first moment estimate of the gradients
$\mathbf{V}_t \leftarrow \beta_2 \mathbf{V}_{t-1} + (1-\beta_2) {g_t}^2$: Update the second moment estimate of the gradients
$\hat{\mathbf{M}}_t \leftarrow \frac{\mathbf{M}_t}{(1-\beta_1^t)}$: Bias correction of first moment estimate
$\hat{\mathbf{V}}_t \leftarrow \frac{\mathbf{V}_t}{(1-\beta_2^t)}$: Bias correction of second moment estimate
$\boldsymbol{\omega}_t \leftarrow \boldsymbol{\omega}_{t-1} - \alpha \frac{\hat{\mathbf{M}}_t}{(\sqrt{\hat{\mathbf{V}}_t} + \epsilon)}$: (Update parameters; ϵ is a small constant to prevent 0 denominator)
end while
return $\boldsymbol{\omega}_t$

In the applications in Section 6, we use the default values for $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\alpha = 0.001$ as proposed by (Kingma and Ba (2014)), and we set $\epsilon = 10^{-7}$, which is the default value in `TensorFlow` package from `R`.

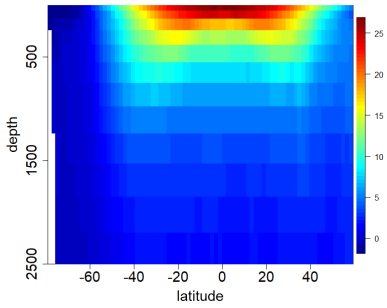
S2 Visual Comparisons

This Section provides Supplement to Section 6.1.2, where we compare the performance of our V3N with PPGP under the two test scenarios specified in Section 6.1.1.1 using the UVic ESCM model data.

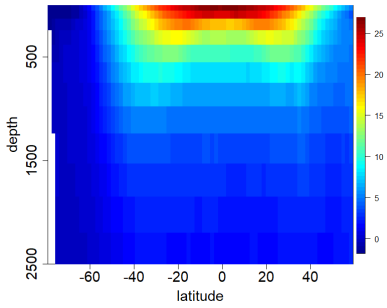
S2.1 Visual Comparisons under the First Case Test Scenario



Original model output

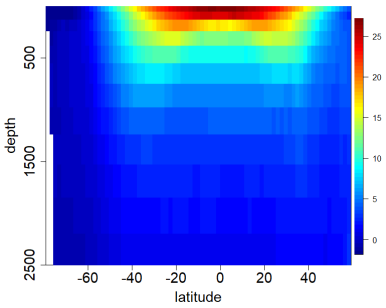


Emulated output via V3N

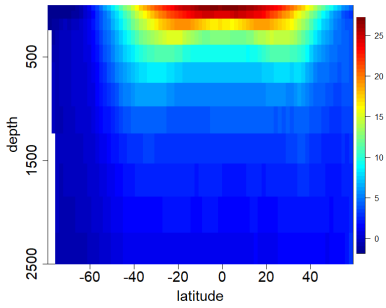


Emulated output via PPGP

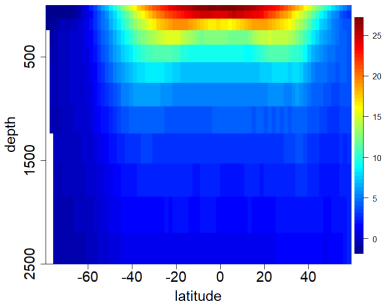
Fig. S1: An example UVic ESCM output (top) and the emulated output using V3N (bottom left) and PPGP (bottom right).



Original model output



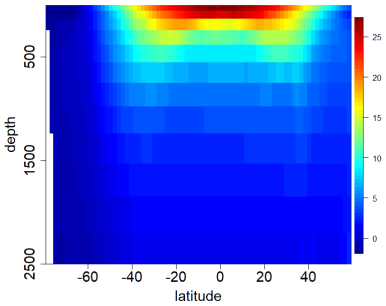
Emulated output via V3N



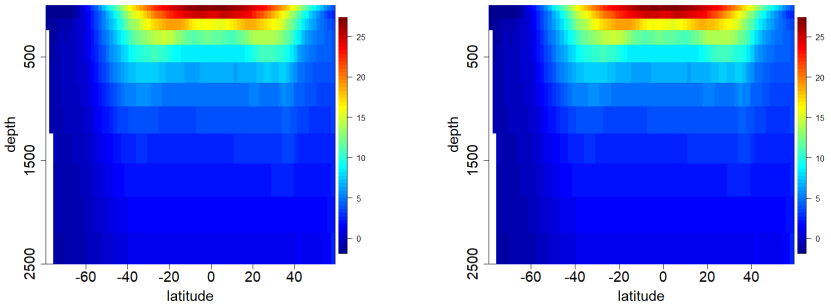
Emulated output via PPGP

Fig. S2: Same as Figure S2.1 with the model output at another parameter setting.

S2.2 Visual Comparisons under the Second Case Test Scenario



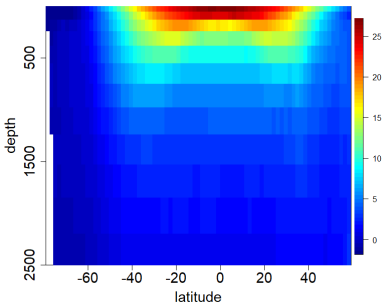
Original model output



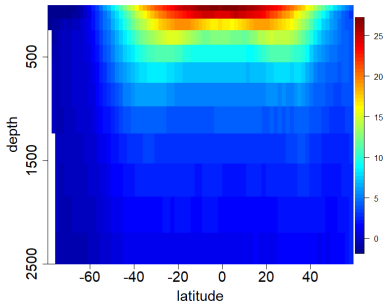
Emulated output via V3N

Emulated output via PPGP

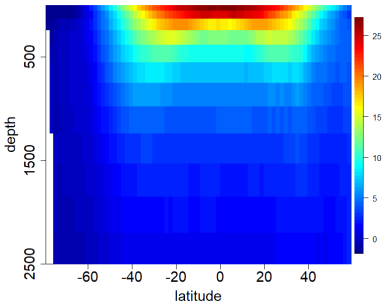
Fig. S3: Same as Figures S2.1 and S2.2 with the model output at another parameter setting.



Original model output



Emulated output via V3N



Emulated output via PPGP

Fig. S4: Another visual comparison between the computer model output (top) and the emulated outputs from our method (bottom left) and PPGP (bottom right).

S3 Model Diagnosis

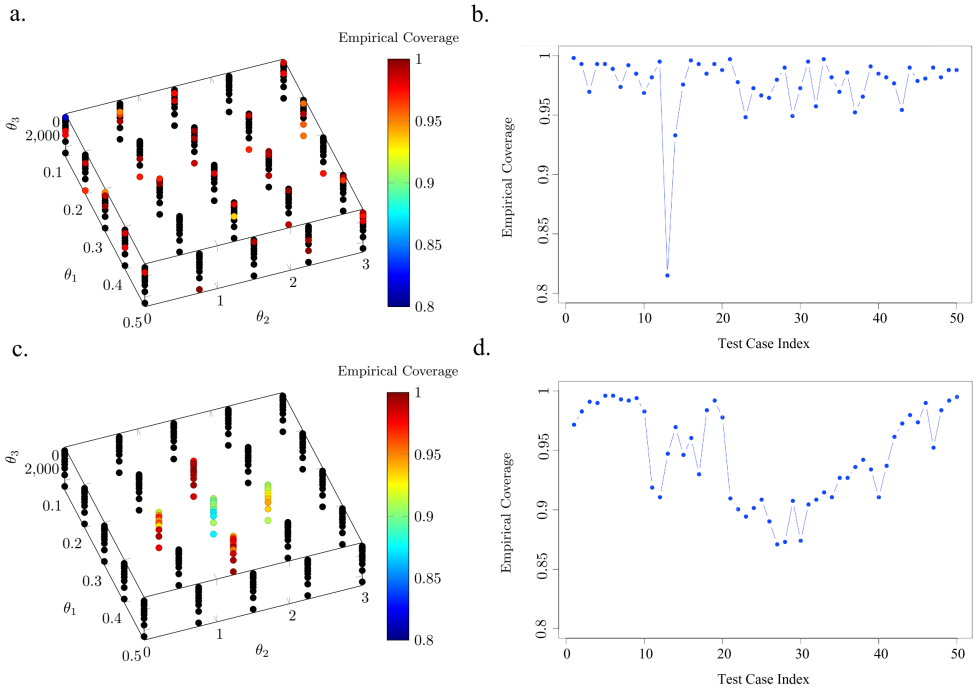


Fig. S5: (a) Empirical coverage values depending on parameter settings in the *short-range* scenario in the UVic ESCM example. (b) Empirical coverage values depending on the test case index in the *short-range* scenario in the UVic ESCM example. (c) Empirical coverage values depending on parameter settings in the *long-range* scenario in the UVic ESCM example. (d) Empirical coverage values depending on the test case index in the *long-range* scenario in the UVic ESCM example.

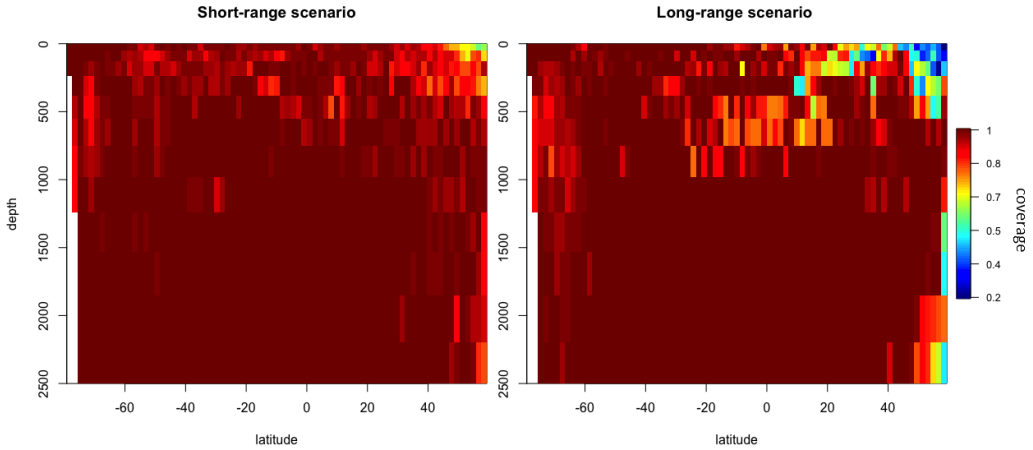


Fig. S6: Empirical coverage values depending on spatial locations in the *short-range* scenario (a) and the *long-range* scenario in the UVic ESCM example

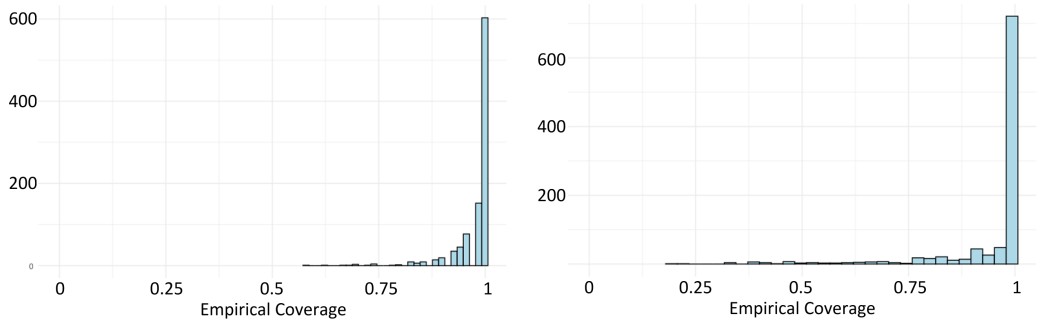


Fig. S7: Distribution of empirical coverage values in the *short-range* scenario (a) and the *long-range* scenario in the UVic ESCM example

S4 Incorporating Spatial Dependence in Prediction Intervals

We have tried two different approaches to incorporate spatial dependence in the prediction intervals generated by our method: the spatial conformal prediction (SCP) method by Mao, Martin, and Reich (2024) and fixed rank kriging (FRK) by Cressie and Johannesson (2008). For each approach, we first generate the predictions $\hat{Y}_{i1}^*, \ldots, \hat{Y}_{in}^*$ for the target parameter setting $\boldsymbol{\theta}_i$ using V3N and then apply the method (SCP or FRK) to generate the spatially correlated intervals for all spatial locations.

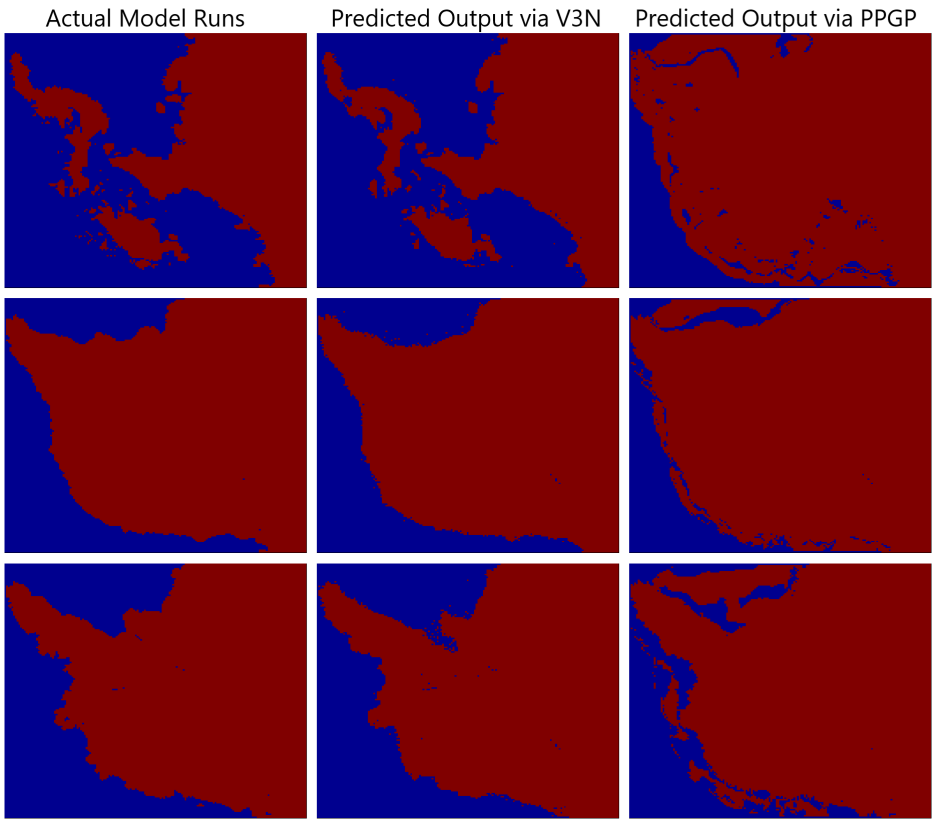


Fig. S8: Same as Figure 6 in the document, but for the dichotomized patters at 0. Blue pixels represent 0 or negative values and dark red pixels represent positive values.

For SCP, we use the `R` code provided by the authors (https://github.com/mhuiying/scp) and set the bandwidth parameter to 0.1 after a grid search, except for one test case that resulted in a numerical error. For that case, we used the bandwidth parameter of 0.2. For FRK, we use the `R` package `FRK` (Zammit-Mangion & Cressie, 2021), discretizing the spatial domain into 10000 cells as the Basic Areal Units (BAUs) and setting the number of basis-function resolutions to 3. The results are summarized in Tables S1 and S2, which show that the use of SCP or FRK results in well-calibrated prediction intervals, with empirical coverages close to the nominal coverage. However, the lengths of the prediction intervals become much larger than those of the original prediction intervals from MC dropout for each location.

Table S1: A comparison of the uncertainty quantification performance of SCP and FRK in the *short range* scenario

Method	EC_{PI} (95%)	L_{PI}
SCP	0.9555	0.4643
FRK	0.9394	0.6334

Table S2: A comparison of the uncertainty quantification performance of SCP and FRK in the *long range* scenario

Method	EC_{PI} (95%)	L_{PI}
SCP	0.9540	0.4343
FRK	0.9399	0.6370