A Addition information of the method

A.1 Additive noise to ocean states

We assume the target values have Gaussian noise coming from different measurements. We manually added noise with reasonable scales to each of the state variable. The following table shows the variances of the added Gaussian noise.

Table 2: Variances of the Gaussian noise

Variables	Salinity	Temperature	Meridional Velocity	Zonal Velocity
σ^2	0.0001	0.25	0.0025	0.0025

We assume the variance of the values at each spatial locations are indepdent, i.e., the covariance matrix of the values in the domain is a diagonal matrix. More specifically, if we assume the data follows a multivariate Gaussian distribution, the mean functions is collectively determined by the neural network output while its covariance matrix is a diagonal matrix with elements determined by the neural network as well, shown in (1).

$$p(\mathbf{y}|\mathbf{x},\kappa) = \mathcal{N}(\mathbf{y}|\boldsymbol{\mu}_\theta(\mathbf{x},\kappa), \sigma^2_\theta(\mathbf{x},\kappa)\mathbf{I}), \tag{1}$$

where $\mathbf{x}, \mathbf{y}$ are the state variables at current and next time step, κ is the perturbed parameter that affects the evolution of the dynamics, $\boldsymbol{\mu}_\theta$ and $\boldsymbol{\sigma_\theta}$ are the mean and variance functions parameterized by the neural network, and $\mathbf{I}$ is an identity matrix.

A.2 Ensemble FNOs

We follow the framework in deep ensemble [15] and train a collection of FNOs independently to construct the ensembles. Let M denote the number of models in the ensemble. Then for each model, the prediction is denoted as

$$\hat{\mathbf{y}}_m = \mathcal{G}_{\theta_m}(\mathbf{x},\kappa), \quad m = 1, 2, \ldots, M. \tag{2}$$

In this work, we take the equal-weighted ensemble prediction for the given $\mathbf{x}$ and κ, which is defined as

$$\hat{\mathbf{y}} = \frac{1}{M}\sum_{m=1}^{M} \hat{\mathbf{y}}_m. \tag{3}$$

As we train each ensemble member to perform heteoscedastic regression, we can decompose the predictive uncertainty into aleatoric and epistemic following the decomposition of total variance.

$$\mathrm{Var}[y] = \underbrace{\mathbb{E}[\mathrm{Var}[y|x]]}_{\text{aleatoric uncertainty}} + \underbrace{\mathrm{Var}[\mathbb{E}[y|x]]}_{\text{epistemic uncertainty}} \tag{4}$$

we use sample unbiased estimators to calculate the uncertainties as shown in Sec. 3.

A.3 Loss functions

With N data points $\{(\mathbf{x}_i, \kappa_i), \mathbf{y}_i\}_{i=0}^{N-1}$ in the training set, the Gaussian negative log-likelihood function is defined as (5), where k is the number of dimensions in the data. The negative log-likehood is shown as follows.

$$\begin{aligned} -\log p(\mathbf{Y}|\mathbf{X}, \kappa, \theta) &= \sum_{i=0}^{N-1} \frac{k}{2} \log(2\pi\sigma_{\theta,i}^2) + \frac{(\mathbf{y}_i - \boldsymbol{\mu}_{\theta,i})^\top (\mathbf{y}_i - \boldsymbol{\mu}_{\theta,i})}{2\sigma_{\theta,i}^2} \\ &= \sum_{i=0}^{N-1} \frac{k}{2} \log(\sigma_{\theta,i}^2) + \frac{(\mathbf{y}_i - \boldsymbol{\mu}_{\theta,i})^\top (\mathbf{y}_i - \boldsymbol{\mu}_{\theta,i})}{2\sigma_{\theta,i}^2} + \text{const.} \end{aligned} \tag{5}$$

With this formulation, we can easily adopt the Gaussian distribution implementation in ML frameworks and output the negative log-probability of given data loss and train the network.

The second loss is the quantile loss, defined in 6. Specifically, we aim to train the network to predict the 0.1587, 0.5, and 0.8413 quantile levels.

$$\mathcal{L} = \begin{cases} q(\hat{y} - y), & \hat{y} \geq y \\ (1 - q)(y - \hat{y}), & \hat{y} < y \end{cases} \tag{6}$$

This allows us to use the associated quantile values to compute the mean and variance of the Gaussian distribution, shown in (7) and (8).

$$\mu = Q(0.5) \tag{7}$$

$$\sigma^2 = \left(\frac{Q(0.8413) - Q(0.1587)}{2}\right)^2 \tag{8}$$

where Q is the quantile function.

A.4 Ensemble selections

We use top-K (Alg. 1) and greedy selection (Alg. 2) criteria to construct ensembles.

Algorithm 1: Top-K Selection

Input: Ensemble size K, model candidates list M
Output: Ensemble E containing K members
$M \leftarrow$ `SortModels`(M) ; `// Sort the models based on HPO objectives`
$E \leftarrow M[:K]$;
return E;

Algorithm 2: Greedy Selection with Replacement

Input: Ensemble size K, initial size k, model candidate list M
Output: Ensemble E containing K members

```
M ← SortModels(M) ;            // Sort the models based on HPO objectives
Initialize E ← M[: k];
while size(E) < K do
    L ← EmptyList();
    for m in M do
        E_temp ← Append(E, m) ;    // Adding one member at at time
        l = ComputeLoss(E_temp) ;   // Compute ensemble prediction loss
        L ← Append(L, l) ;         // Append loss to the loss list
    i ← Argmin(L) ;       // Obtain the index of the model to be added
    E ← Append(E, M[i]) ;       // Add the selected model to the ensemble
return E;
```

A.5 Hyperparameter search details

We list the hyperparameters in this work in Tab. 3. We utilized 32 computing nodes and 4 GPUs per node to excecute the HPO search. In total, we finished 993 evaluations.

Table 3: Hyperparameter search space of this work, largely adopted from [26]. We include the choice of loss function between GNLL and PB loss as additional hyperparameter.

Variable Names.	Type	Range/Choice	Explanation
`padding`	`bool`	`[True, False]`	If zero-pad the data.
`padding_type`	`str`	`['constant', 'reflect', 'replicate', 'circular']`	Types of padding.
`coord_feat`	`bool`	`[True, False]`	If use domain coordinates as additional features.
`lift_act`	`str`	`['relu', 'leaky_relu', 'prelu', 'relu6', 'elu', 'selu', 'silu', 'gelu', 'sigmoid', 'logsigmoid', 'softplus', 'softshrink', 'softsign', 'tanh', 'tanhshrink', 'threshold', 'hardtanh', 'identity', 'squareplus']`	Activation function for lifting layers. The choices include common activation functions implemented in `PyTorch`.
`num_FNO`	`int`	[2, 16]	The number of FNO blocks.
`num_latent_feat`	int	[2, 64]	The number of latent features in FNO blocks. This is equivalent to the number of channels in an image representation.
`num_modes`	`int`	[2, 32]	The number of Fourier modes to keep.
`num_proj_layers`	`int`	[2, 16]	The number of projection layers.
`proj_size`	`int`	[2, 16]	Projection layer size.
`proj_act`	`str`	`['relu', 'leaky_relu', 'prelu', 'relu6', 'elu', 'selu', 'silu', 'gelu', 'sigmoid', 'logsigmoid', 'softplus', 'softshrink', 'softsign', 'tanh', 'tanhshrink', 'threshold', 'hardtanh', 'identity', 'squareplus']`	Activation function for projection layers. The choices include common activation functions implemented in `PyTorch`.
`alpha`	`float`	(0, 1)	Weight associated with MSE and negative ACC in the loss function.
`optimizer`	`str`	`['Adadelta', 'Adagrad', 'Adam', 'AdamW', 'RMSprop', 'SGD']`	Types of optimizers.
`loss_fn`	`str`	`['nll', 'quantile']`	Choice of the loss function
`lr`	`float`	(1e-6, 1e-2)	Learning rate
`weight_decay`	`float`	(0, 0.1)	The weighting factor of the L_2 regularization.
`batch_size`	`int`	(2, 64)	The batch size of training data during training.

B Dataset

We use the perturbed parameter SOMA dataset adopted from [27] which contains 100 independent forward simulations. Each simulation produced 30 days of the ocean dynamics influenced by the parameter κ. The original dataset contains

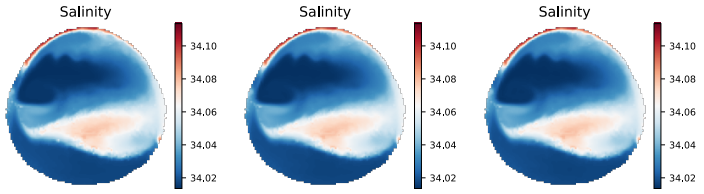
17 ocean variables in a three-dimensional basin. For computational efficiency and providing proof of concept, we reduced the data to two dimensions by selecting four prognostic variables (salinity, temperature, meridional and zonal velocity) at the sea surface. The 2D domain is near-circular, and for each variable in the domain, the data shape is of (100, 100). So the data instances contain areas outside the near-circular domain. We padded these areas with zero in this work.

C More results

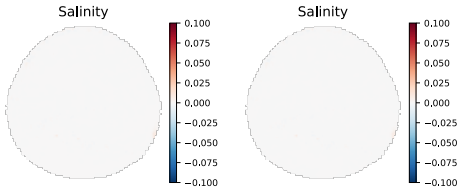
C.1 Mean prediction visualizations

Table 4: Ensemble single-step **mean** forecast R^2(↑)

	Top-K Ensemble	Greedy Ensemble
Salinity	0.9982	0.9983
Temperature	0.9997	0.9994
Meridional Velocity	0.9998	0.9991
Zonal Velocity	0.9999	0.9996

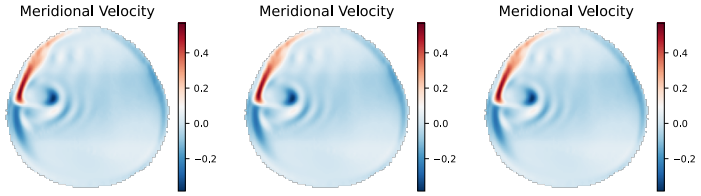


(a) True (b) Top-K Ensemble (c) Greedy Ensemble

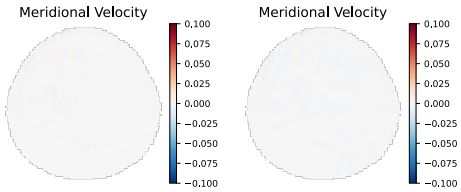


(d) Top-K difference (e) Greedy Ensemble

Fig. 5: Example true and predicted mean salinity state from the ensembles.

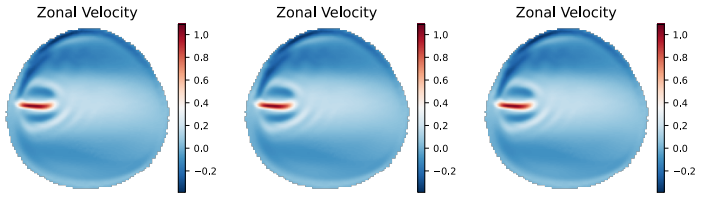


(a) True (b) Top-K Ensemble (c) Greedy Ensemble

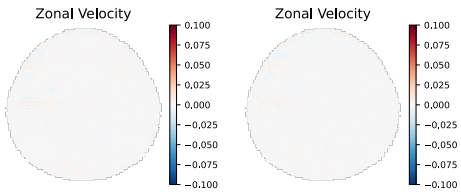


(d) Top-K difference (e) Greedy Ensemble

Fig. 6: Example true and predicted mean meridional velocity from the ensembles.



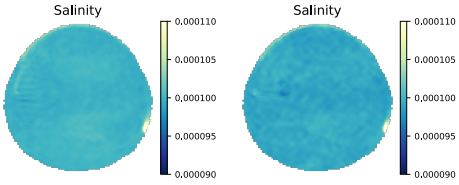
(a) True (b) Top-K Ensemble (c) Greedy Ensemble



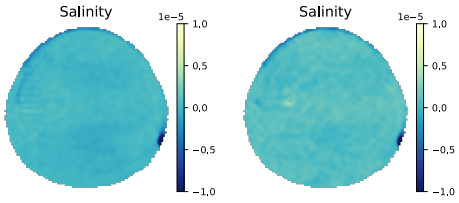
(d) Top-K difference (e) Greedy Ensemble

Fig. 7: Example true and predicted mean zonal velocity from the ensembles.

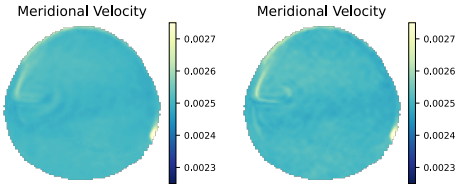
C.2 Aleatoric uncertainty visualizations



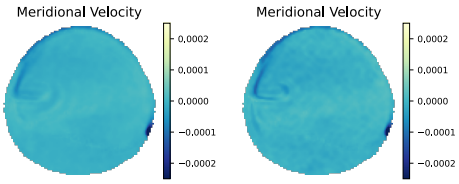
(a) Top-K σ^2_{al} (b) Greedy σ^2_{al}



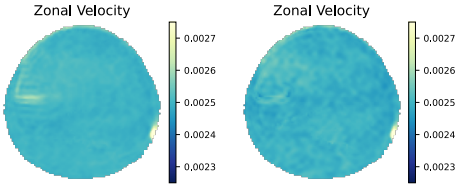
(c) Top-K σ^2_{al} error (d) Greedy σ^2_{al} error



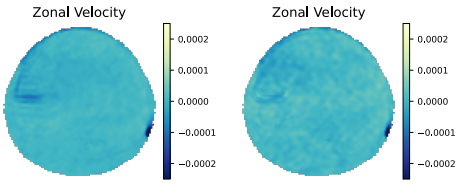
(a) Top-K σ^2_{al} (b) Greedy σ^2_{al}



(c) Top-K σ^2_{al} error (d) Greedy σ^2_{al} error

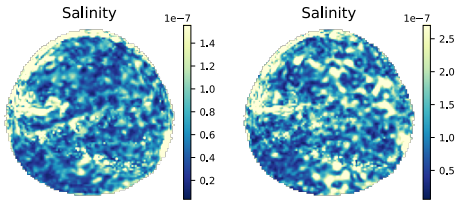


(a) Top-K σ^2_{al} (b) Greedy σ^2_{al}

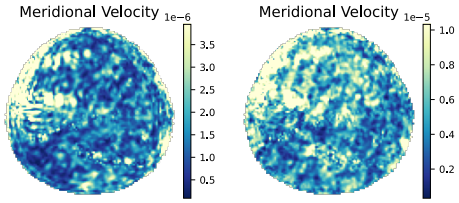


(c) Top-K σ^2_{al} error (d) Greedy σ^2_{al} error

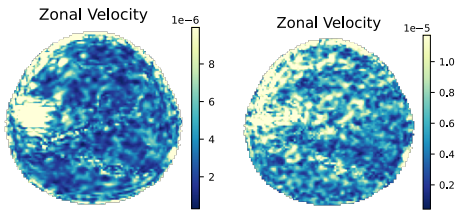
C.3 Epistemic uncertainty visualization



(a) Top-K σ^2_{ep} (b) Greedy σ^2_{ep}



(a) Top-K σ^2_{ep} (b) Greedy σ^2_{ep}



(a) Top-K σ^2_{ep} (b) Greedy σ^2_{ep}