Disentangled Uncertainty with Hyperparameter Deep Ensembles in Ocean Modeling

Yixuan Sun[1], Romain Egele[2], and Prasanna Balaprakash[2]

1- Argonne Nat. Lab., USA; 2- Oak Ridge Nat. Lab., USA

Abstract.

Gaussian negative log-likelihood (GNLL) is a standard loss function for quantifying uncertainty in data with additive Gaussian noise, but its optimization with stochastic gradient descent (SGD) often results in instability and suboptimal solutions. This work proposes replacing GNLL with the pinball loss (PB), or quantile loss, as a more robust alternative during SGD. In the context of ocean modeling with Fourier Neural Operators (FNOs), we perform hyperparameter optimization (HPO), treating both GNLL and PB as candidate losses for aleatoric uncertainty estimation. Models identified during HPO are used to construct ensembles for forecasting and epistemic uncertainty quantification. PB loss consistently outperforms GNLL in HPO, yielding models with more accurate aleatoric uncertainty. Ensembles built from these models provide disentangled aleatoric and epistemic uncertainties, achieving stable rollouts up to 30 days with epistemic uncertainty reliably predicting forecast errors. These results highlight PB loss as a superior alternative to GNLL for uncertainty quantification and emphasize the value of hyperparameter ensembles for robust long-term forecasting.

1 Introduction

Oceans play an important role in mitigating climate change by absorbing and transporting carbon dioxide. Modeling oceans using first-principle-based simulators is time-consuming and computationally expensive, making it challenging for tasks such as optimization, sensitivity analysis, and uncertainty quantification. Deep learning models are promising surrogate models to accelerate computation and improve scalability. However, the standard deep learning models are point estimators and fail to provide uncertainty estimates, potentially leading to overconfident predictions. These uncertainties can arise from at least two sources [1]: the aleatoric uncertainty (inherent to the data $P(Y|X)$), and, the epistemic uncertainty (inherent to the model estimation). When the conditional distribution of the data $P(Y|X)$ follows a Gaussian distribution, a simple yet effective way to quantify both uncertainties in deep learning is to train an ensemble of models using GNLL loss during SGD [2]. However, optimizing GNLL loss has stability issues and results in suboptimal performance [3, 4]. To address this, this work explores PB loss as a drop-in replacement of the NLL loss. We first optimize the hyperparameters or our model and then build an ensemble from these explored models. Our results demonstrate that the PB loss consistently outperforms the GNLL loss. The ensembles then provides accurate predictions in both single-step forecasts and rollouts up to 30 days while also offering quantified epistemic uncertainty.

2 Related work

Deep learning-based ocean modeling has emerged in recent years [5]. Much effort has been devoted to modeling spatiotemporal ocean properties [6, 7]. However, these frameworks require fine-tuning and do not provide estimated unceraínty. This work builds upon FNOs for ocean modeling [8, 6, 9] and constructs ensembles of FNOs via HPO for streamlined forecasting and uncertainty quantification.

Uncertainty quantification in deep learning models has been widely discussed [1, 10]. The goal is usually to estimate two types of uncertainty: aleatoric (from the data) and epistemic (from the model). Aleatoric uncertainty, in the case of a Gaussian additive noise, is typically modeled by the conditional variance $V[Y|X]$ where X, Y are two random variables representing the input and target respectively. The base model can estimate this quantity with an adapted loss function such as GNLL [11, 12, 13]. On the other hand, epistemic uncertainty corresponds to uncertainty in model estimation is often addressed through direct Bayesian modeling [14] or discrete approximations such as deep ensembles [15, 16]. Our work aligns both with the literature on HPO and deep ensembles. One of the most widely used HPO algorithms is Bayesian optimization [17], and many frameworks [18, 19, 20] have been designed to provide Bayesian optimization-based hyperparameter search capabilities. In this study, we use DeepHyper [21, 22], a framework that offers out-of-the-box efficient parallel HPO and ensemble selection methods with disentangled uncertainties.

3 Methods

The problem we aim to solve is to learn from noisy data of the true dynamics of a wind-driven baroclinic ocean model and quantify disentangled uncertainties in our predictions. The dynamics are treated as an initial value problem. The solution at time t is represented as $\mathbf{x}_t = \mathbf{x}_0 + \int_0^t f(\mathbf{x}(\tau, \kappa))d\tau$, where κ is an external parameter influencing the state evolution. We use FNOs, $\mathcal{G}_\theta$, parameterized by θ to approximate the solution operator. In particular, with Markovian assumption, we train the FNOs to approximate the solution at $t+1$ given the state at t. Therefore, the training data have input-output pairs of $\{(\mathbf{x}_t, \kappa), \mathbf{x}_{t+1}\}$.

We assume aggregated measurement errors from different sources and add synthetic Gaussian noise to the future state in the data pairs following the literature [23]. Now, the target states follow $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \kappa) = \mathcal{N}(\mu(\mathbf{x}_t, \kappa), \sigma(\mathbf{x}_t, \kappa))$. We use FNOs to approximate the mean and standard deviation of the distributions, $[\mu_\theta || \sigma_\theta] = \mathcal{G}_\theta(\mathbf{x}_{t+1}, \kappa)$, where $||$ is the concatenation operator. The de-facto loss to train such models is GNLL [2]. However, as mentioned earlier, GNLL carries stability issues due to high impact from the gradient of the predictive variance [4]. We explore PB loss [24] as an alternative to train the models to predict 0.1587, 0.5, and 0.8413 quantile levels to also estimate the mean and variance. Such exploration is executed in the HPO process where **the objectives include GNLL but the loss function for SGD is an hyperparameter between GNLL and PB losses**.

With the completed HPO results, we form the ensembles in two ways. With the preset numbers of members in the ensemble $K = 10$, the first way immediately selects the top-K best performing models in the HPO results and equally-weighted form the ensemble. The second way is based on simplified greedy selection [25], where we start with k initial models ($k = 1$ in this work) and add models one at a time according to the best ensemble performance until reaching K models in the ensemble. The ensemble predictions are computed as $\hat{\mu} = \frac{1}{K}\sum_{i=1}^{K} \mu_{\theta_i}$, where θ_i represents the ith member (hyperparameter and parameters) in the ensemble. Using the law of total variance, the aleatoric uncertainty of the ensemble is computed as the average variance across members, $\hat{\sigma}^2_{\text{al}} = \frac{1}{K}\sum_{i=1}^{K} \sigma^2_{\theta_i}$ and the epistemic uncertainty of the ensemble is the variance of predictions across members, $\hat{\sigma}^2_{\text{ep}} = \frac{1}{K}\sum_{i=1}^{K}(\mu_{\theta_i} - \hat{\mu})^2$ [2]. The total uncertainty just being the sum of both $\hat{\sigma}^2 = \hat{\sigma}^2_{\text{al}} + \hat{\sigma}^2_{\text{ep}}$. We provide supplementary materials[1] containing more details of the loss functions, HPO process, and model selection criteria.

4 Numerical experiments

We used the preprocessed perturbed parameter Simulation Ocean Mesoscale Activity (SOMA) dataset [9], which includes four state variables (salinity, temperature, meridional velocity, and zonal velocity) at the sea surface. The dataset comprises 100 independent simulations with varying parameter settings. Each simulation generates a trajectory of states over 30 time steps (days). We split the data based on the independent simulations for training and evaluation, training the models to produce single-step forecasts as described in Sec. 3. Additional details about the dataset are provided in App. B.

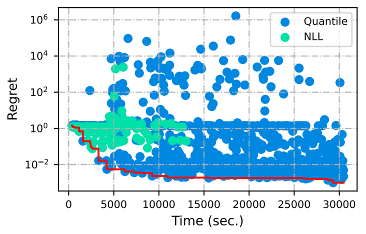


Fig. 1: HPO simple regret with PB loss in blue and GNLL loss in green.

Using the data with noisy targets, we followed the HPO procedure detailed in [26], with the additional inclusion of the loss function choice between NLL and PB loss, to create the candidate pool for constructing the ensembles. Fig. 1 shows the model performance regret (score difference from the best performing model) during the search process. With NLL and Mean Squared Error (MSE) as objectives, the search gradually favored PB loss, which resulted in the best-performing models. This highlights the advantage of using PB loss to train heteroscedastic regression models over NLL, leading to more stable training and accurate predictions.

[1]Supplementary materials at https://github.com/iamyixuan/ocean_ensemble/blob/main/esann25/ESANN25_with_supp.pdf

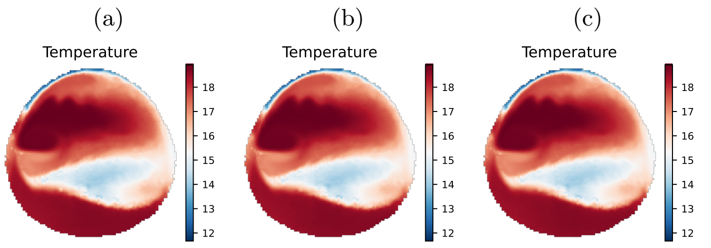


Fig. 2: Example true (a) and predicted mean temperature state (b)(c) from the ensembles.

Table 1: Single-step ensemble forecast performance

(a) **mean** forecast $NMAE(\downarrow)$

	Top-K Ensemble	Greedy Ensemble
Salinity	5.2448×10^{-6}	5.3811×10^{-6}
Temperature	6.6908×10^{-4}	7.5983×10^{-4}
Meridional Velocity	1.6075×10^{-2}	1.9485×10^{-2}
Zonal Velocity	9.5705×10^{-3}	9.0555×10^{-3}

(b) **aleatoric** uncertainty $NMAE$ $(\downarrow)$.

	Top-K Ensemble	Greedy Ensemble
Salinity	4.3630×10^{-3}	9.7198×10^{-3}
Temperature	5.6226×10^{-3}	7.8888×10^{-3}
Meridional Velocity	5.2449×10^{-3}	6.1723×10^{-3}
Zonal Velocity	4.7132×10^{-3}	6.1094×10^{-3}

To quantify predictive uncertainties, we formed two ensembles using the top-K and greedy selection criteria. Both ensembles produced one-step forecasts and longer-term rollouts with quantified uncertainties based on the ensemble predictions. For both ensembles, we set the number of members to $K = 10$.

In single-step forecasts, both ensembles accurately predict the state variables across the domain. Tab. 1a shows the ensemble performance on all four state variables in terms of normalized mean absolute error (NMAE), defined as $NMAE = \sum |y - \hat{\mu}| / \sum y$, y being the true states (noiseless) and $\hat{y}$ the ensemble predictions. A lower $NMAE$ indicates better performance. The ensembles have marginal performance difference regarding the mean prediction where the top-K ensemble slightly outperforms the greedy ensemble in three out of the four variables. We observe that the top-K ensemble better captures the variance of the data distribution, shown in Tab. 1b. To visualize the ensemble predictions, we show an instance of the temperature state as an example. Fig. 7 shows the ensemble predictions on temperature profile where the ground truth and predictions are visually identical.

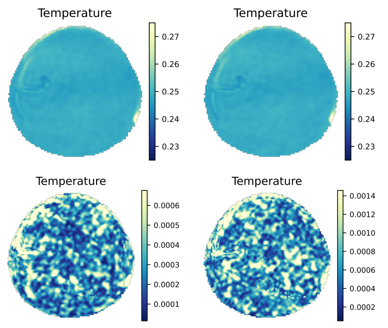


Fig. 3: Temperature aleatoric (top row) and epistemic uncertainties (bottom row). Left column shows uncertainty from the top-K ensemble and right column the greedy ensemble.

Regarding the quantified uncertainties, shown in Fig. 3, both ensembles accurately capture the true aleatoric uncertainty (0.25). Meanwhile, both ensembles present small epistemic uncertainties, indicating high degree of agreement among

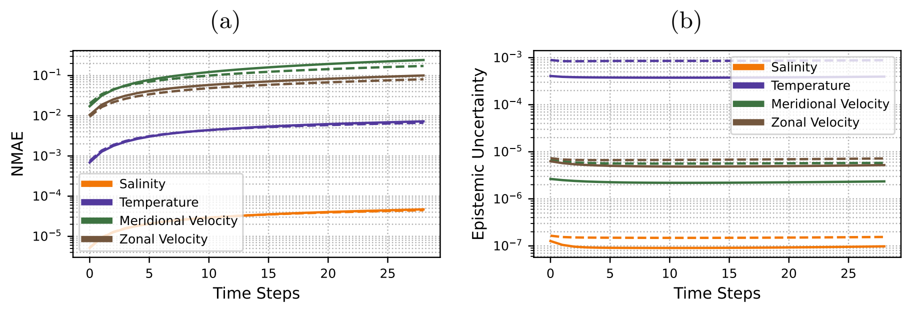


Fig. 4: Ensemble rollout trajectory-averaged $NMAE$ scores (a) and epistemic uncertainty (b) with time steps (days). Solid lines represent the top-K ensemble and dashed lines the greedy ensemble.

the ensemble members.

We further investigated the ensembles' ability of producing stable and accurate autoregressive rollouts. We used nine trajectories, each consisting of 30 time steps (days) for evaluation. We evaluated performance across the spatial domain and reported the trajectory-averaged scores per variable. Fig. 4a shows the ensemble rollout NMAE. Both ensembles produce stable rollout while the greedy ensemble shows slower error accumulation. Regarding the epistemic uncertainty during the rollout, both ensembles present minimal and stable epistemic uncertainty, shown in Fig 4b, indicating little variation in the predictions from the ensemble members. **We observe the growth of epistemic uncertainty becomes visually significant for longer rollouts ($>$ 200 steps)**. We include additional details of the dataset, noise scales, and results in 1.

5 Conclusion

This work leveraged HPO to form ensembles for modeling ocean dynamics using FNOs. By comparing GNLL and PB loss along with other hyperparameters, the HPO identified the PB loss as superior for learning data distribution parameters, leading to better models. Ensembles formed using two criteria accurately captured one-step dynamics and produced stable 30-day rollouts. This approach effectively models ocean dynamics from noisy data while providing robust uncertainty quantification for trustworthy applications.

References

[1] M. Abdar et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information fusion*, 76:243–297, 2021.

[2] R. Egele et al. Autodeuq: Automated deep ensemble with uncertainty quantification. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 1908–1914. IEEE, 2022.

[3] D. Megerle et al. Stable optimization of gaussian likelihoods, 2023.

[4] M. Seitzer et al. On the pitfalls of heteroscedastic uncertainty estimation with probabilistic neural networks, 2022.

[5] M. J. Er et al. Research challenges, recent advances, and popular datasets in deep learning-based underwater marine object detection: A review. *Sensors (Basel, Switzerland)*, 23, 2023.

[6] S. Bire et al. Ocean emulation with Fourier neural operators: Double gyre. *Authorea Preprints*, 2023.

[7] Y. Gou et al. DeepOcean: A general deep learning framework for spatio-temporal ocean sensing data prediction. *IEEE Access*, 8:79192–79202, 2020.

[8] Z. Li et al. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[9] Y. Sun et al. Parametric sensitivities of a wind-driven baroclinic ocean using neural surrogates. In *Proceedings of the Platform for Advanced Scientific Computing Conference*, pages 1–10, 2024.

[10] W. Zhe et al. A survey on uncertainty quantification methods for deep learning, 2024.

[11] C. Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[12] J. Ho et al. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[13] Y. LeCun et al. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

[14] L. Jospin et al. Hands-on bayesian neural networksâa tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, 17(2):29–48, 2022.

[15] B. Lakshminarayanan et al. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

[16] F. Wenzel et al. Hyperparameter ensembles for robustness and uncertainty quantification. *Advances in Neural Information Processing Systems*, 33:6514–6527, 2020.

[17] D. Jones et al. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 1998.

[18] R. Liaw et al. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018.

[19] T. Akiba et al. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.

[20] J. Bergstra et al. Hyperopt: a python library for model selection and hyperparameter optimization. *Computational Science & Discovery*, 8(1):014008, 2015.

[21] P. Balaprakash et al. DeepHyper: Asynchronous hyperparameter search for deep neural networks. In *2018 IEEE 25th international conference on high performance computing (HiPC)*, pages 42–51. IEEE, 2018.

[22] R. Egele. *Optimization of Learning Workflows at Large Scale on High-Performance Computing Systems*. PhD thesis, Université Paris-Saclay, 2024.

[23] A. Wong et al. Argo data 1999–2019: Two million temperature-salinity profiles and subsurface velocity observations from a global array of profiling floats. *Frontiers in Marine Science*, 7:700, 2020.

[24] Y. Chung et al. Beyond pinball loss: Quantile methods for calibrated uncertainty quantification. *Advances in Neural Information Processing Systems*, 34:10971–10984, 2021.

[25] R. Caruana et al. Ensemble selection from libraries of models. In *Proceedings of the twenty-first international conference on Machine learning*, page 18, 2004.

[26] Y. Sun et al. Streamlining ocean dynamics modeling with fourier neural operators: A multiobjective hyperparameter and architecture optimization approach. *Mathematics*, 12(10):1483, 2024.

[27] Y. Sun et al. Surrogate neural networks to estimate parametric sensitivity of ocean models. *arXiv preprint arXiv:2311.08421*, 2023.

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).

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

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 $\boldsymbol{\sigma}_{\boldsymbol{\theta}}$ 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).

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

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

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

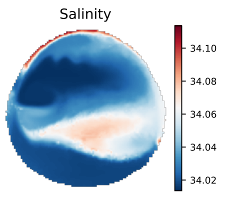
17 ocean variables in a three-dimensional basin. For computational efficiency and providing proof of concept, we reduced the data to two dimensions by selecting four prognostic variables (salinity, temperature, meridional and zonal velocity) at the sea surface. The 2D domain is near-circular, and for 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

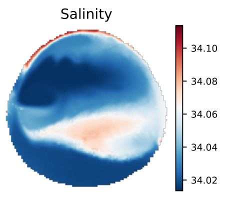
C.1 Mean prediction visualizations

Table 4: Ensemble single-step **mean** forecast $R^2(\uparrow)$

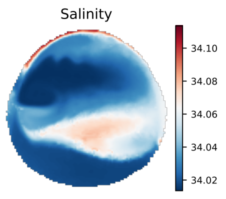
	Top-K Ensemble	Greedy Ensemble
Salinity	0.9982	0.9983
Temperature	0.9997	0.9994
Meridional Velocity	0.9998	0.9991
Zonal Velocity	0.9999	0.9996



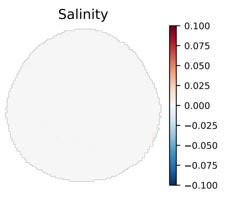
(a) True



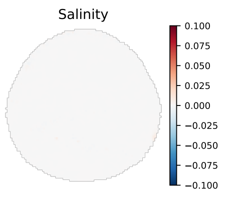
(b) Top-K Ensemble



(c) Greedy Ensemble

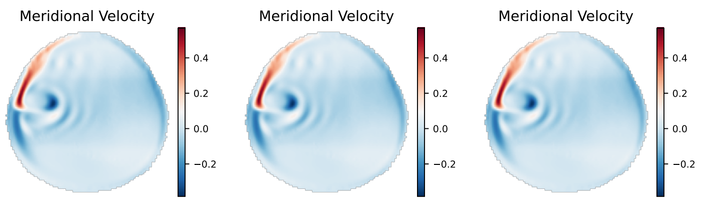


(d) Top-K difference

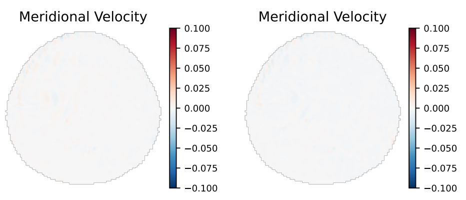


(e) Greedy Ensemble

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

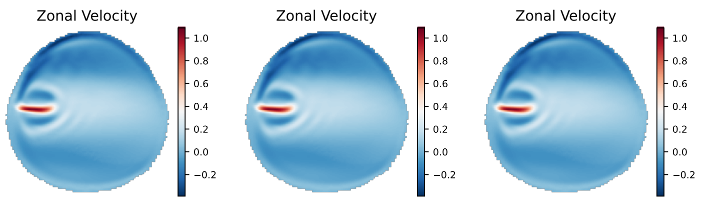


(a) True (b) Top-K Ensemble (c) Greedy Ensemble

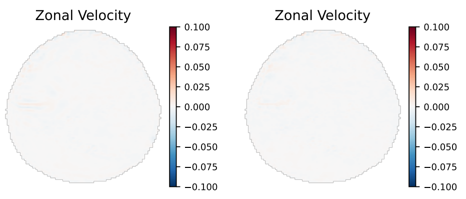


(d) Top-K difference (e) Greedy Ensemble

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



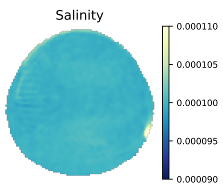
(a) True (b) Top-K Ensemble (c) Greedy Ensemble



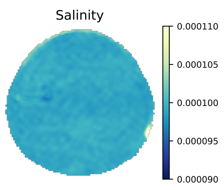
(d) Top-K difference (e) Greedy Ensemble

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

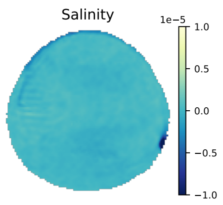
C.2 Aleatoric uncertainty visualizations



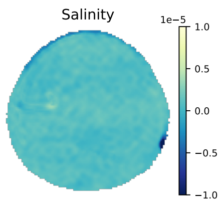
(a) Top-K σ^2_{al}



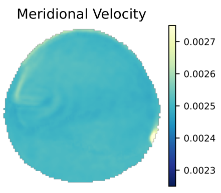
(b) Greedy σ^2_{al}



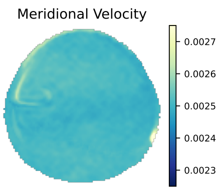
(c) Top-K σ^2_{al} error



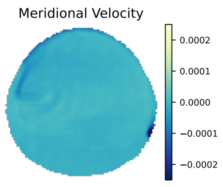
(d) Greedy σ^2_{al} error



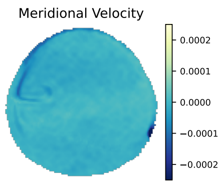
(a) Top-K σ^2_{al}



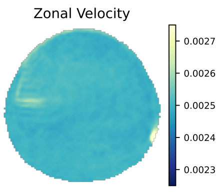
(b) Greedy σ^2_{al}



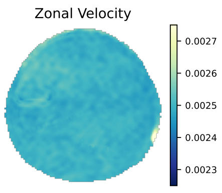
(c) Top-K σ^2_{al} error



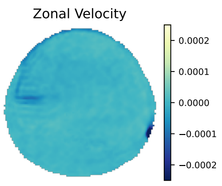
(d) Greedy σ^2_{al} error



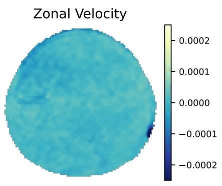
(a) Top-K σ^2_{al}



(b) Greedy σ^2_{al}

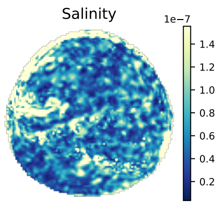


(c) Top-K σ^2_{al} error

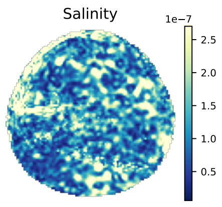


(d) Greedy σ^2_{al} error

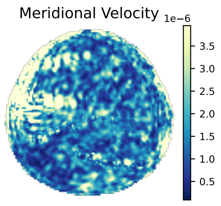
C.3 Epistemic uncertainty visualization



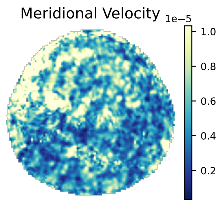
(a) Top-K σ^2_{ep}



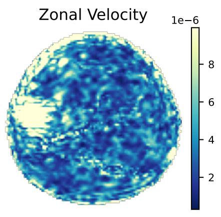
(b) Greedy σ^2_{ep}



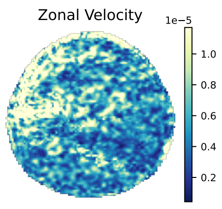
(a) Top-K σ^2_{ep}



(b) Greedy σ^2_{ep}



(a) Top-K σ^2_{ep}



(b) Greedy σ^2_{ep}