






Article

Streamlining Ocean Dynamics Modeling with Fourier Neural Operators: A Multiobjective Hyperparameter and Architecture Optimization Approach

Yixuan Sun ¹, Ololade Sowunmi ², Romain Egele ^{1,3}, Sri Hari Krishna Narayanan ¹, Luke Van Roekel ⁴
and Prasanna Balaprakash ^{5,*}

- ¹ Argonne National Laboratory, Lemont, IL 60439, USA; yixuan.sun@anl.gov (Y.S.); romain.egele@universite-paris-saclay.fr (R.E.); snarayan@mcs.anl.gov (S.H.K.N.)
² Department of Mathematics, Florida State University, Tallahassee, FL 32304, USA; osowunmi@fsu.edu
³ Laboratoire Interdisciplinaire des Sciences du Numérique, Université Paris-Saclay, 91190 Gif-sur-Yvette, France
⁴ Los Alamos National Laboratory, Los Alamos, NM 87545, USA; lvanroekel@lanl.gov
⁵ Oak Ridge National Laboratory, Oak Ridge, TN 37830, USA
* Correspondence: pbalapra@ornl.gov

Abstract: Training an effective deep learning model to learn ocean processes involves careful choices of various hyperparameters. We leverage DeepHyper’s advanced search algorithms for multiobjective optimization, streamlining the development of neural networks tailored for ocean modeling. The focus is on optimizing Fourier neural operators (FNOs), a data-driven model capable of simulating complex ocean behaviors. Selecting the correct model and tuning the hyperparameters are challenging tasks, requiring much effort to ensure model accuracy. DeepHyper allows efficient exploration of hyperparameters associated with data preprocessing, FNO architecture-related hyperparameters, and various model training strategies. We aim to obtain an optimal set of hyperparameters leading to the most performant model. Moreover, on top of the commonly used mean squared error for model training, we propose adopting the negative anomaly correlation coefficient as the additional loss term to improve model performance and investigate the potential trade-off between the two terms. The numerical experiments show that the optimal set of hyperparameters enhanced model performance in single timestepping forecasting and greatly exceeded the baseline configuration in the autoregressive rollout for long-horizon forecasting up to 30 days. Utilizing DeepHyper, we demonstrate an approach to enhance the use of FNO in ocean dynamics forecasting, offering a scalable solution with improved precision.

Keywords: ocean modeling; operator learning; hyperparameter optimization

MSC: 68T20



Citation: Sun, Y.; Sowunmi, O.; Egele, R.; Narayanan, S.H.K.; Van Roekel, L.; Balaprakash, P. Streamlining Ocean Dynamics Modeling with Fourier Neural Operators: A Multiobjective Hyperparameter and Architecture Optimization Approach. *Mathematics* **2024**, *12*, 1483. <https://doi.org/10.3390/math12101483>

Academic Editors: Andreas Lintermann and Guillaume Houzeaux

Received: 29 March 2024

Revised: 6 May 2024

Accepted: 7 May 2024

Published: 10 May 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Deep learning techniques significantly enhance scientific computing, serving as efficient surrogate models for complex, time-intensive first-principles-driven simulations. Their application in climate modeling leverages large amounts of high-resolution, multidimensional data. It addresses the computational challenges posed by increasingly complex climate systems, resulting in remarkable advances in environmental research [1–5].

The oceans play a leading role in the climate system via the storage and transport of anthropogenically generated heat and carbon dioxide, a role that is crucial for mitigating climate change. Simultaneously, oceanic processes that redistribute mass, heat, and salt are instrumental in driving phenomena such as hurricanes, extreme precipitation, and droughts [6]. Deep learning models have empowered ocean modeling from a spatial–temporal forecast of select ocean features [7–9] to parameterization [10–12]. Despite

the success of those models, however, selecting and fine-tuning appropriate model hyperparameters remain challenging, often requiring a significant amount of manual search via trial and error.

Hyperparameters are the non-trainable parameters determining the data preprocessing, network structure, and training procedures; and they highly impact the performance of deep learning models [13]. Searching for a proper combination that works optimally for the specific problem can be challenging. With the increasing complexity of performant deep learning models, manual tuning conducted in a trial-and-error way becomes infeasible, given that hyperparameter configurations exist within a vast search space.

Deep learning models for ocean processes must accurately capture the mean behavior of the ocean state profile and account for the detailed variation throughout the domain. In modeling climate processes using deep learning models, one typically formulates the problem as an image-to-image regression problem, where the pixels of an image represent the state values within the domain of interest and the image channels are associated with different state variables. One often uses per-pixel mean squared error (MSE) as the loss function because of differentiability and correspondence to maximized likelihood with assumed Gaussian errors to train these models [3,5,14]. However, MSE penalizes large errors more significantly and usually produces images in favor of averaged pixel values that may be blurry and unnatural [15,16]. This situation might become problematic in producing ocean dynamics forecasts because the values are relatively stable and have a long time scale compared with the weather, where the values evolve closely to the mean field. Many modifications and different loss functions have been proposed to mitigate the issue of MSE [15,17], but they are highly problem-dependent and can be complex. Here, specifically for ocean modeling with deep neural networks, we propose using the negative value of a simple metric, the anomaly correlation coefficient (ACC) [18], commonly used in climate model evaluation, as the additional loss term on top of the MSE to address this potential issue.

We construct the model with Fourier neural operators (FNOs) [19] and utilize DeepHyper [20,21] to perform hyperparameter and architecture searches to automate the model selection of FNO for an idealized baroclinic wind-driven ocean system. The aim is to predict four prognostic variables at a single time step in the future, given the variable values and a physical parameter at the current time. We also anticipate an improvement in the autoregressive rollout performance, which typically requires special treatment during training [5,14,22]. This expected enhancement is due to the selected model's ability to provide more accurate forecasts for single-time steps. We investigate the potential disadvantages of pure MSE loss and the effect of combining MSE and negative ACC with an associated hyperparameter, weighing the importance of the two. Furthermore, we have expanded the hyperparameter optimization (HPO) across a high-performance computing (HPC) environment involving more than 20 nodes, each equipped with four GPUs, while integrating effective early stopping criteria to refine the search process, yielding more timely and potent outcomes. The numerical experiments show that using the optimal hyperparameter configuration from the search results in a model that outperforms the baseline in the single time-stepping forecast of all four prognostic variables and significantly improves the model's autoregressive performance for a long-range rollout. We expect this work to pave the way for a more systematic model construction process for ocean/climate modeling using FNOs and their variants.

The contribution of this paper is threefold: (1) We propose a multiobjective hyperparameter optimization approach for FNOs in ocean modeling, leveraging DeepHyper's advanced scalable search algorithms. (2) We introduce the negative anomaly correlation coefficient as an additional loss term to improve model performance and investigate the potential trade-off between the two terms. (3) We demonstrate the effectiveness of the proposed approach in enhancing the performance of FNOs in ocean dynamics forecasting in both single-stepping prediction and longer horizon autoregressive forecasting. The rest of this paper is organized as follows. Section 2 introduces the work involving deep

learning studies for ocean modeling, FNO, and the general hyperparameter search processes with multiple objectives. Section 3 formulates the problem into an operator learning setting and describes Bayesian optimization with multipoint acquisitions for parallelization. Sections 4 and 5 define the hyperparameter search space and discuss the search results and the impact of hyperparameters associated with data preprocessing, neural architecture, and training processes. Section 6 summarizes our conclusions and briefly mentions future directions.

2. Related Work

2.1. Fourier Neural Operator

The FNO and its variants [19,23–25] have shown an outstanding ability to solve partial differential equations and predictive tasks in various scientific domains [26,27]. In particular, the FNO powered climate modeling and enabled accurate mid- and long-range prediction at a fraction of the cost compared with first-principles-based simulations [5,28]. The general idea behind the FNO is to composite convolutional kernel integrals in the Fourier domain to approximate the solution operator. The FNO generally has three major components: lifting layers, Fourier operator blocks, and projection layers. During lifting, the network transforms the input functions (e.g., initial condition or other parameters) to higher-dimensional spaces. The transformed input goes through a series of FNO blocks where the fast Fourier transform (FFT) transforms the data, followed by convolutional operations in the frequency domain. A preset number of high-frequency modes in the results are filtered out, preserving general features and smoother behaviors associated with lower-frequency modes. An inverse FFT then transforms the data back to the physical domain. Finally, the projection components map the data back to their original dimensions, forming the solution. As an operator learning framework, the FNO learns the mapping between function spaces and can predict beyond the resolution of the training data. Therefore, they are suitable for solving problems involving partial differential equations and various mesh representations of the domain, including weather and ocean modeling.

2.2. Hyperparameter Optimization with DeepHyper

DeepHyper [21] is a Python package that provides asynchronous parallel hyperparameter and neural architecture search algorithms. It provides parallel implementations of the SMAC (sequential model-based algorithm configuration) algorithm [29], which is a type of Bayesian optimization algorithm [30]. For parallelization, it can leverage different backends with minimal code changes, such as threads, processes, and MPI. We will now provide an overview of the hyperparameter optimization algorithms that we used in this work.

2.2.1. Centralized Bayesian Optimization

Bayesian optimization (BO), also known as efficient global optimization [30], is a popular approach for hyperparameter optimization frameworks because of its fast convergence properties. BO is a type of black-box optimization algorithm where the core idea is to iteratively update an internal surrogate model in order to minimize the number of direct queries to the real “expensive” black-box function. Compared with grid search and random search, BO is efficient in exploring the high-dimensional hyperparameter space due to its ability to model the relationship between the hyperparameters and the objectives and provide a better trade-off between exploration and exploitation in selecting the next hyperparameter configuration leading to the most expected improvement in model performance [31]. The standard BO algorithm is sequential, thus making it difficult to leverage parallel computing resources available on HPC systems. Within DeepHyper, BO can be parallelized through a centralized or decentralized architecture. In the centralized scheme, a single manager decides the function evaluation locations when workers conduct the evaluations in parallel. In contrast, the decentralized scheme equips workers with their own optimizer and allows all operations in parallel. The latter provides maximum scalability [32] when performing large-scale hyperparameter optimization (i.e., >1000 parallel function evaluations). In our

case, we use the centralized approach with a q UCB (q -Upper Confidence Bound) acquisition function [33] for the multipoint acquisition strategy. This method provides better scalability than the usual constant-liar multipoint acquisition strategy [34], also known as Kriging Believer [35].

In a centralized architecture, the central coordinator determines the next hyperparameters at which the function should be tested and then distributes these tasks to available remote workers. The surrogate model used is Extremely Randomized Trees [36], a type of random forest approach [37] that provides better epistemic uncertainty estimates thanks to a random-split strategy when building tree nodes.

2.2.2. Multiobjective Optimization

The most common objective in hyperparameter optimization for neural networks is to minimize the validation loss. In our problem setup, the loss function contains two terms, MSE and negative ACC, and we plan to investigate any potential trade-off between the two. MSE accounts for the average behavior of the state variable fields, penalizing large errors, whereas ACC focuses on anomalies between forecast and true fields, emphasizing pattern nuances. In ocean modeling, we not only care about accurate forecasts of the mean field over time, for example, the gyre circulation, but also are interested in the precise characterization of the local variations, for example, intergyre exchange from eddies. We investigate the relationship between the two objectives through the lens of multiobjective optimization.

A trade-off between MSE and ACC would mean that on the optimal frontier, the Pareto front, one objective cannot be improved without degrading the other. On the contrary, without a trade-off, both objectives can be improved jointly, thus providing a dominant point on the Pareto front [38], that is, in the objective space. A simple way to find the Pareto optimal solutions is to combine the objectives and perform optimization only over the combined single objective, known as scalarization [39]. With a trade-off between objectives, however, this process corresponds to optimizing for a fixed trade-off. DeepHyper can perform multiobjective hyperparameter optimization through randomized scalarization within Bayesian optimization [40]. Randomized scalarization resamples random objective weights for each new suggestion, thus reducing the multiobjective optimization problem to a single-objective optimization problem where each weight vector represents a fixed trade-off between objectives. By resampling different weights, the Pareto-Front can be explored. We leverage this capability to optimize for both validation MSE and ACC.

2.3. Data-Driven Ocean Modeling

Conventional first-principles-based simulations of ocean evolution, while being reliable and accurate, are computationally intensive and can become intractable for tasks such as uncertainty quantification, sensitivity analysis, and optimization. With the advance of deep learning models and the computational efficiency once trained, data-driven methods for modeling ocean dynamics have emerged [41,42]. Some recent developments address spatial-temporal ocean property forecasts [43], information mining for remote sensing [44], rainfall pattern segmentation over oceans [45], and spatial-temporal ocean sensing data prediction [7]. However, based on the specific application scenarios and datasets, building and training these models effectively usually involves careful design of the model architecture and significant trial and error to fine-tune model training to achieve a better overall performance. This process introduces a new round of computational overhead. To address this issue, we show a workflow that leverages FNOs and hyperparameter optimization to streamline the modeling process of ocean dynamics, which other data-driven models can easily adapt.

3. Method

3.1. Problem Formulation

The dynamics of the idealized baroclinic wind-driven ocean model can be formulated as an initial value problem (IVP), which starts with the initial state, x_0 , and the model

parameter(s), κ . We aim to solve for the state $\mathbf{x}(t)$ (i.e., the state prognostic variables) at time t . The IVP, $d\mathbf{x}(t, \kappa)/dt = f(\mathbf{x}(t))$, with $\mathbf{x}(0) = \mathbf{x}_0$, leads to the solution at time t , $\mathbf{x}_t = \mathbf{x}_0 + \int_0^t f(\mathbf{x}(\tau, \kappa))d\tau$. The solution at the discrete time step $t + 1$ can be expressed as

$$\mathbf{x}(t + 1) = \mathbf{x}(t) + \int_t^{t+1} f(\mathbf{x}(\tau, \kappa))d\tau. \tag{1}$$

For simplicity, we use the subscript t to denote the current time from this point. We use an FNO, \mathcal{N}_θ , parameterized by learnable weights and biases θ to approximate the solution operator to (1). In particular, we trained the neural network to approximate the following mapping,

$$\mathbf{x}_{t+1} = \mathcal{N}_\theta(\mathbf{x}_t, \kappa), \tag{2}$$

where \mathbf{x} is the state variable vector, t represents the current time, and κ is the parameter that impacts the trajectories of state variables. With the training dataset \mathcal{D} containing input–output pairs, the learning objective is to minimize the empirical loss function,

$$\begin{aligned} \mathcal{L}(\theta, \mathcal{D}) &= \alpha \cdot \mathcal{L}_{\text{MSE}} + (1 - \alpha) \cdot \mathcal{L}_{\text{NegACC}}, \\ \mathcal{L}_{\text{MSE}} &= \frac{1}{N} \sum_{i=1}^N \|\mathbf{x}_{t+1}^{(i)} - \mathcal{N}_\theta(\mathbf{x}_t^{(i)}, \kappa^{(i)})\|_2^2, \\ \mathcal{L}_{\text{NegACC}} &= -\frac{\sum_{i=1}^N (\mathcal{N}_\theta(\mathbf{x}_t^{(i)}, \kappa^{(i)}) - \bar{\mathbf{x}}_{t+1})(\mathbf{x}_{t+1}^{(i)} - \bar{\mathbf{x}}_{t+1})}{\sum_{i=1}^N \sqrt{(\mathcal{N}_\theta(\mathbf{x}_t^{(i)}, \kappa^{(i)}) - \bar{\mathbf{x}}_{t+1})^2 (\mathbf{x}_{t+1}^{(i)} - \bar{\mathbf{x}}_{t+1})^2}}, \end{aligned} \tag{3}$$

where N is the number of data points in the training set, and \bar{x} is the average value of the state variables. The loss function contains two terms: the standard MSE loss and the negative ACC. We assign a weighting factor α , treated as a hyperparameter, to adjust the relative importance between the two terms. We expect the trained neural network to predict the state variables one step forward accurately.

3.2. Bayesian Optimization and Multipoint Acquisition

Bayesian optimization is an efficient global optimization method for black-box functions with the presence of noise. Our HPO problem focuses on finding a set of hyperparameters that maximize the preset objectives. The black-box function, $f(x)$, in this context, is the function that maps hyperparameters to the objectives. Formally, we aim to solve the following,

$$\max_p \{f(p) : p = (p_D, p_N, p_T) \in \mathcal{P}\}, \tag{4}$$

where p is a set of hyperparameters and subscripts D , N , and T represent hyperparameters in the data preprocessing, neural architecture, and training process, respectively. The hyperparameter search space \mathcal{P} is predefined. The evaluation of $f(p)$ is computationally expensive because it requires complete training of neural networks. Therefore, a surrogate model is needed to approximate these expensive functions, allowing for a more efficient exploration of the hyperparameter space. With the surrogate model, \mathcal{M} , in place, BO follows the general formulation. \mathcal{M} models the conditional distribution of the objective values of a given hyperparameter configuration, $q(f(p)|D)$, where D is the current evaluation of the objective function. BO selects the next hyperparameter configuration, p' , based on the values of an acquisition function, accounting for the potential objective improvement. The acquisition function balances the trade-off between exploration and exploitation of the hyperparameter space.

We used the default surrogate model in the DeepHyper random forest [37], to perform BO. Random forest regressors have scaling advantages over more commonly used Gaussian processes [32].

In BO, the acquisition function describes the quality of the input and determines the next evaluation points. We used the upper confidence bound (UCB), defined as the follows,

$$UCB(p) = \mu(p) + c \cdot \sigma(p), \quad (5)$$

where $\mu(p)$ and $\sigma(p)$ are the mean and standard deviation of the surrogate predictions, respectively. The constant, c , controls the trade-off between exploration and exploitation. BO iteratively evaluates $f(p)$ and determines the next evaluation point(s) p' based on the value of $UCB(p)$.

To perform efficient searches and leverage the HPC environment, we enabled multipoint acquisition with UCB to parallelize BO. This poses a multipoint optimization problem. In the centralized scheme, a manager performs BO and selects hyperparameter configurations, and workers evaluate the configurations and return the results back to the manager. Since BO typically updates the surrogate model and selects points sequentially, we used the q UCB strategy [33] to ensure effective evaluation point selection in parallel, where various values of c are drawn from the exponential distribution for the UCB acquisition function, and different next points are chosen according to these values to achieve an optimal balance between exploration and exploitation.

4. Numerical Experiments

This section describes the numerical experiment setup, including the data generation, hyperparameter search space, and optimization execution.

4.1. Dataset

We adopted and simplified the data generated from [46]. The dataset used in the experiment consisted of 100 simulations of salinity, temperature, zonal velocity, and meridional velocity at the ocean surface of an idealized baroclinic wind-driven ocean model. They were obtained by running the Simulating Ocean Mesoscale Activity (SOMA) test case [47] within a circular basin of a 150 km wide and 100 m deep shelf. An ensemble of simulations was developed by varying the bolus diffusivity (κ_{GM}) in the Gent–McWilliams parameterization. The diffusivity was uniformly sampled from [200, 2000]. Each variable of interest has two dimensions in space, spanning 100 grid points per direction. Therefore, along with κ_{GM} , the data obtained from each simulation (associated with a different κ_{GM}) have the shape of (30, 100, 100, 5), where 30 indicates 30 time steps (days), and 5 includes the four state variables and κ_{GM} . To train models that take the state variables and parameters at the current time and predict the same set of state variables at the next step, we split 30 time steps into 29 input–output pairs per simulation, resulting in 2900 input–output instances in total. We further split the data based on independent simulations into training, validation, and testing sets with a ratio of 0.6, 0.2, and 0.2, respectively. The search for the optimal set of hyperparameters was determined by the model performance on the validation set. The final evaluations were carried out using the testing set.

4.2. Hyperparameter Search Space

The hyperparameters involved in this work mainly belong to three categories: data preprocessing, neural architecture related, and training process related. The neural architecture related hyperparameters determine the hypothesis (model) space, bounding the expressivity and generalizability. The training-process-related hyperparameters affect the learning, of which the goal is to find the optimal model by varying its trainable weights to minimize the loss function over the training and validation data. We focus on searching for the optimal set of hyperparameters, listed in Table 1, that contribute to the best-performing FNO for ocean modeling.

Table 1. List of hyperparameters and their ranges. (a) shows the hyperparameters related to data preprocessing. (b) shows the hyperparameters determining the neural architecture. (c) describes the ones in the training process.

(a)			
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.
(b)			
Variable Names	Type	Range/Choice	Explanation
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.
(c)			
Variable Names	Type	Range/Choice	Explanation
alpha	float	(0, 1)	Weight associated with MSE and negative ACC in the loss function.
optimizer	str	['Adadelata', 'Adagrad', 'Adam', 'AdamW', 'RMSprop', 'SGD']	Types of optimizers.
lr	float	$(10^{-6}, 10^{-2})$	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.

Since the convolution operations in the Fourier domain are the critical components of the FNO, the usage of padding and coordinate features is essential to the learning results. The first set of hyperparameters is whether we use padding for the input data and the padding type. Paddings affect the output of the convolutions at the domain boundaries [48] and potentially mitigate the edge effect introduced by the subsequent FFT. Furthermore,

we consider incorporating the coordinate features on the input data level, allowing the model to learn the explicit solution dependency on the spatial position.

In the lifting block of FNO, which maps the input data to higher dimensions elementwise (preserving the spatial dimension), we aim to optimize the number of layers and the activation functions. The next component in FNO is the operator learning block, consisting of the forward FFT, convolution operation in the frequency domain, filtering out of high-frequency modes, followed by nonlinear activations, and then an inverse FFT. Here, we search for the optimal number of operator learning blocks, the number of Fourier (low-frequency) modes to keep, and the choice of the activation function. The last component in the typical FNO is the projection block, transforming the lifted and convolved data back to the original data dimension. Similar to the lifting block, we are interested in finding the optimal number of projection layers, the number of neurons per layer, and the choice of the activation function.

During training, the choice of the optimizer, batch size, magnitude of the learning rate, weights associated with the loss terms, and regularization could highly impact the quality of the trained model. Therefore, we aim to find an optimal combination of these values that leads to the most accurate model.

4.3. Objectives

We set up the hyperparameter optimization problem to optimize two objectives. The first objective is to minimize the validation MSE, defined as the first term in (3). MSE is calculated pixel-wise, aiming to reach the average minimum value. The second objective is to maximize the validation anomaly correlation coefficient (ACC), defined as the second term in (3), which is a commonly used metric in climate studies. Given that adding negative ACC could potentially avoid the drawbacks of MSE, we treat them as two separate objectives with equal importance and investigate the relationship between ACC and MSE. In DeepHyper, all the searches are focused on maximizing objectives. Therefore, we implemented the objectives as negative MSE and positive ACC.

4.4. Implementation

We constructed the FNO model using the `Nvidia Modulus` package (version 0.4.0) and set up the hyperparameter search space, the black-box function for BO, and the evaluator in `DeepHyper`. The searches used 20 computing nodes with four `Nvidia A100` GPUs per node, on the `Polaris` cluster at the Argonne Leadership Computing Facility. The searches were executed for 6 h which completed approximately 500 evaluations.

Training the FNO with high-dimensional data can be computationally expensive. Therefore, we adopted two stoppers to accelerate the searches and the parallelization. A stopper terminates the training of a neural network based on certain criteria, improving search efficiency. The first stopper used a constant predictor, a constant number that minimizes the MSE term in the loss function (3). In this case, the constant predictor becomes the mean value of the target. The second stopper keeps track of the running time for each training epoch. In our optimized training regimen for each hyperparameter configuration, we implemented an early stopping mechanism for efficiency. Specifically, we terminated the training of any models whose configurations yielded results inferior to those of the constant predictor within the initial 10 epochs or taking over 100 s to complete a single epoch during training. If neither stopper was triggered, the training was terminated at epoch 30, returning the objective values. This approach strategically reduces unnecessary computational expenditure on unpromising model configurations.

With the search results, we conducted a full training (100 epochs) using the set of hyperparameters that resulted in the best objectives. We used the default hyperparameter configuration preset in `Modulus` as a baseline to compare the testing performance with the fully trained model.

5. Results and Discussion

To demonstrate the positive impact of using negative ACC as the additional loss term, we trained models using the default hyperparameters provided by Modulus for 100 epochs using loss functions that are pure MSE and the equally weighted ones of MSE and negative ACC; while the search objectives were validation MSE and ACC, we use Relative Squared Error (RSE), $RSE = \sum(\mathbf{x}_{t+1} - \mathcal{N}_\theta(\mathbf{x}_t, \kappa))^2 / \sum(\mathbf{x}_{t+1} - \bar{\mathbf{x}}_{t+1})^2$, and $1 - ACC$ in their log scale to highlight performance differences. We also use these metrics to evaluate search results in the later sections.

Table 2 shows the $\log(RSE)$ and $\log(1 - ACC)$ values from the two models. Overall, the model trained with the composite loss function achieved lower values for both metrics. Notably, training the network with the composite loss improved MSE values over the pure MSE loss in salinity, meridional velocity, and zonal velocity. These results validate adding negative ACC as part of the loss function. The model training for the subsequent hyperparameter optimization adopts this composite loss.

Table 2. Model performance on the testing set using the baseline models trained with MSE loss and composite loss. The downward arrows indicate the lower values are better.

	$\log(RSE) \downarrow$		$\log(1 - ACC) \downarrow$	
	MSE Loss	MSE + NegACC Loss	MSE Loss	MSE + NegACC Loss
Salinity	−2.202	−2.498	−2.503	−2.800
Temperature	−3.696	−3.061	−4.015	−3.363
Meridional V.	−1.958	−2.067	−2.258	−2.842
Zonal V.	−2.248	−2.303	−2.549	−2.808

5.1. First Objective: Validation Mean Square Error

Varying separate hyperparameters results in differing responses in the validation MSE. Figure 1 shows the parallel coordinate plots of the hyperparameters in the search space, divided into three categories. The curves are ranked based on the objective values in the log scale, where the lighter blue represents favorable configurations. Figure 1a suggests that among the data-related hyperparameters, in this case, the padding (its type and coordinate features, if used) has an unclear effect on the final model performance in validation MSE. The reason could be that the region of interest is a circular basin represented in the regular grids, with the area outside the basin being zeros, and using padding or different types of padding only expands the area, not affecting learning. Figure 1b shows the effect of various training strategies. Among the hyperparameters for training, the loss term weighing factor α near the two ends of its range results in suboptimal performance. A smaller batch size corresponds to a lower validation MSE, while the magnitude of the learning rate seems to have negatively affected the validation MSE. We suspect this was due to the limited number of epochs in the search, where models update more slowly with lower learning rates and wind up with worse performance at the end of training. The weight decay does not show an evident correlation with the model performance. A smaller batch size is considered useful for improving the generalization error and accelerating convergence [49], coinciding with the search results. The configurations with lower validation MSE are present mostly with the AdamW optimizers, while Adadelata is associated with mostly lower-ranked configurations. The Adam optimizer [50], the default optimizer in our cases, has led to mixed model performances. Regarding the neural architecture-related hyperparameters, we can observe evident impacts from the choice of the number of latent channels, number of FNO blocks, number of Fourier modes, number of projection layers, and projection activation functions. More specifically, latent channels being greater than two and less than ten, fewer FNO blocks, a higher number of Fourier modes, and a mid- to low-level number of projection layers correspond to a lower validation MSE. Among the projection activations, the commonly used rectified linear unit (ReLU) led to

a relatively suboptimal performance. However, some ReLU variants—leaky ReLU and PReLU—positively influenced the minimization of the validation MSE.

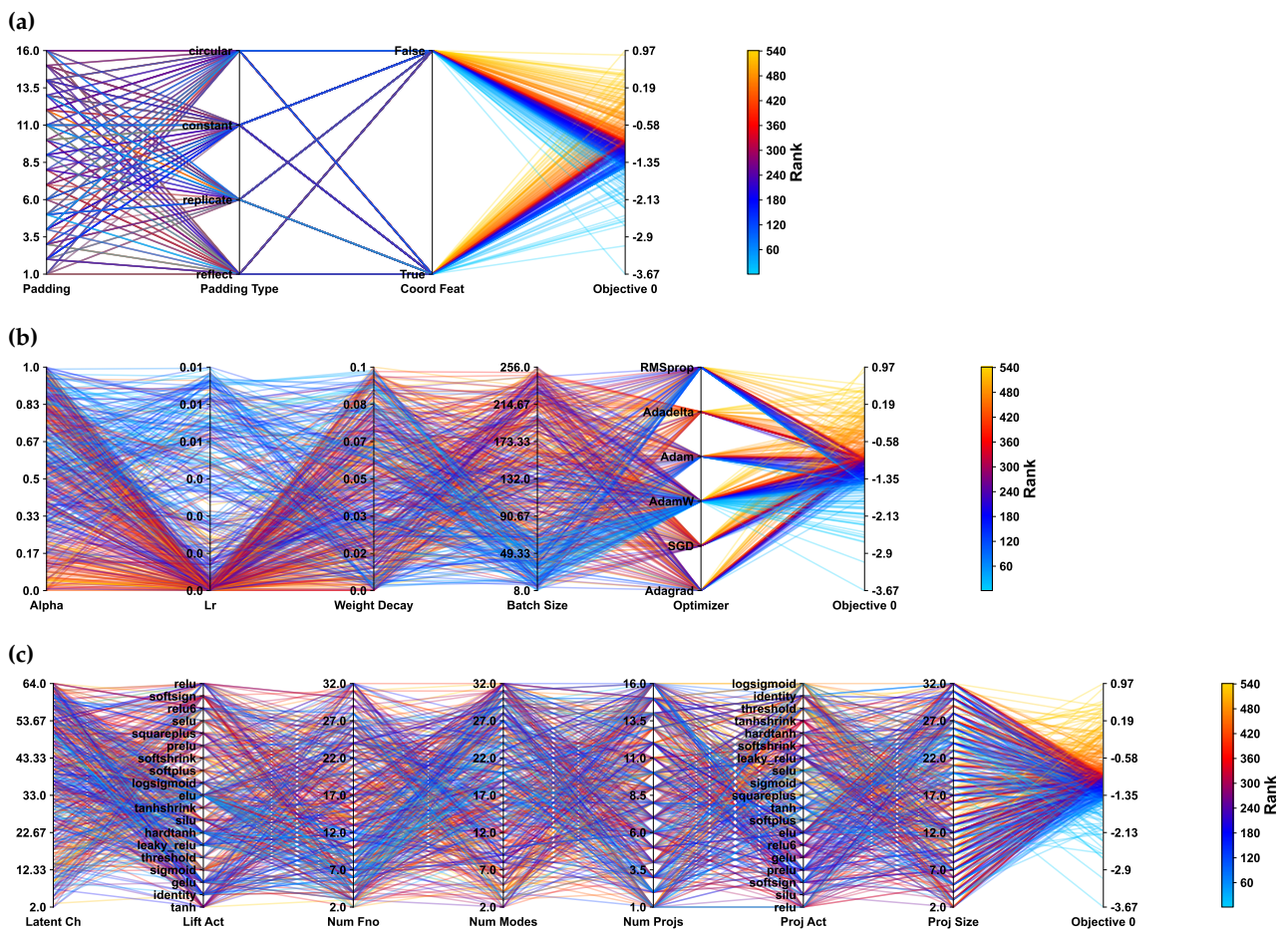


Figure 1. Parallel coordinate plots of the hyperparameters in the search space with respect to the validation MSE (in log scale). The space is divided into three categories. (a) shows the data-related hyperparameters, (b) shows training-related hyperparameters, and (c) contains neural architecture-related hyperparameters.

5.2. Second Objective: Validation Anomaly Coefficient Correlation

The second search objective was the validation ACC, which measures how much the forecasted deviation from the mean field is correlated with the true deviation. A higher ACC is preferred; thus, the search aimed to maximize the ACC. Figure 2 shows the parallel coordinate plots of the impact of hyperparameters on the validation ACC, where the lines are ranked with decreasing ACC values. Different hyperparameters have a different impact. Figure 2a implies that the data preprocessing approaches do not have an obvious impact on the validation ACC, which coincides with the case for validation MSE. In Figure 2b, the loss weighing factor α does not present an obvious impact on the validation ACC. Similar to the impact on the validation MSE, a greater learning rate positively affects the ACC. The influence of the weight decay magnitude is clearer compared with it on the validation MSE, where a lower weight decay leads to higher ACC. Conversely, the impact of the batch size is not as evident as it is for the validation MSE, but a smaller value continues to have a positive impact on the performance. Regarding the choice of optimizers, the AdamW optimizer leads to the most high-ranking configurations. Regarding the neural architecture-related hyperparameters, shown in Figure 2c, a higher number of latent channels is associated with higher ACC, a trend that is less evident in the validation MSE. Regarding the activation functions in the lifting layer, elu seems to be most helpful, but the

advantage is not prominent. Similar to the impact on the validation MSE, fewer FNO blocks lead to a better ACC. Within each FNO block, the number of Fourier modes to keep does not suggest a straightforward pattern, where having around 20 such nodes results in many highly ranked configurations. The rest of the hyperparameters determining the neural architecture do not present a strong correlation with the validation ACC.

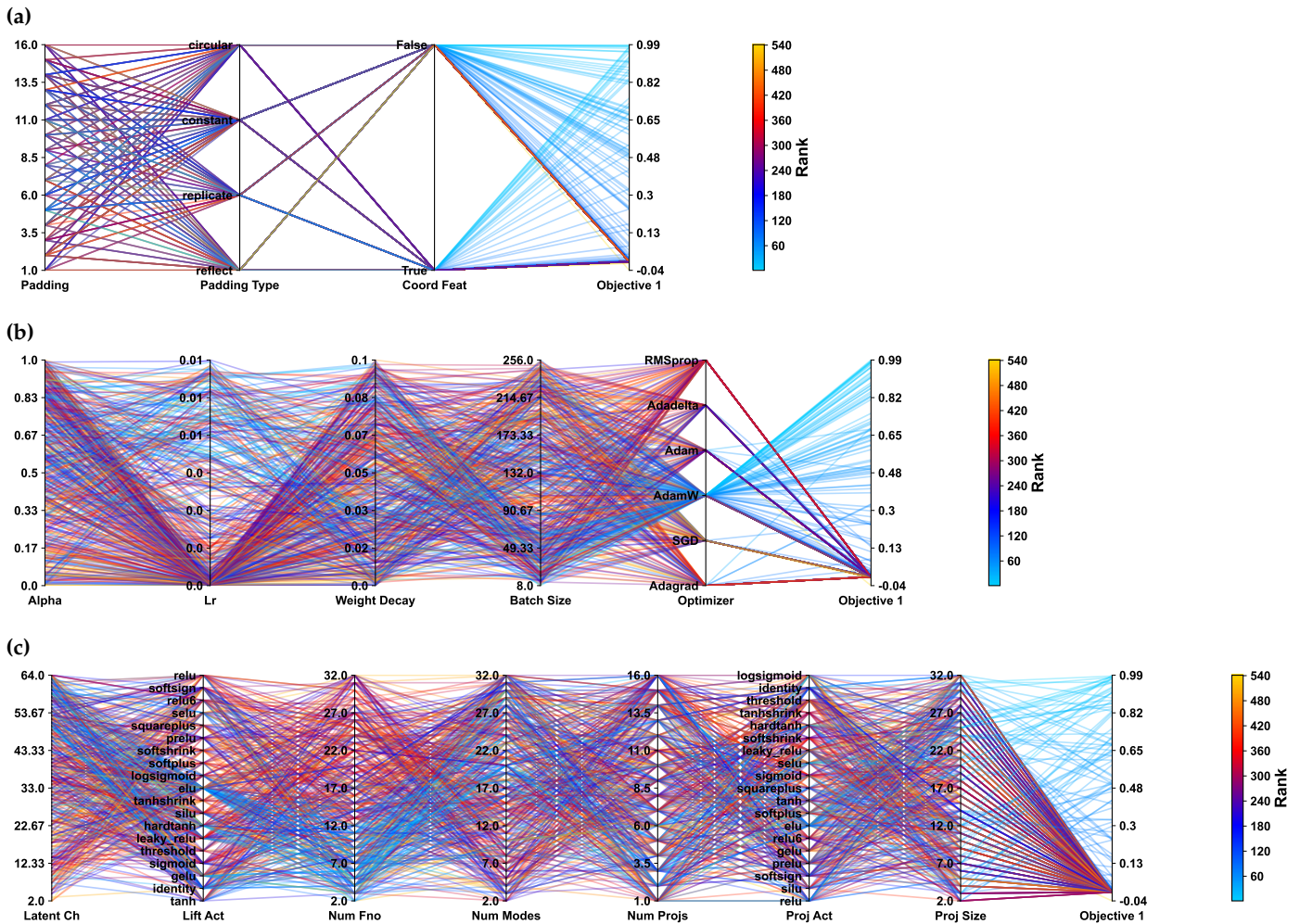


Figure 2. Parallel coordinate plots of the hyperparameters in the search space with respect to the validation ACC. The space is divided into three categories. (a) shows the data-related hyperparameters, (b) shows training-related hyperparameters, and (c) contains neural architecture-related hyperparameters.

5.3. Relationship between MSE and ACC

In the search configuration, we sought to optimize for both lower validation MSE and higher ACC. Given that the composite loss improves learning compared with MSE-only loss, we want to determine whether the two loss terms have any trade-offs. Figure 3 shows the scatter plot of the MSE and negative ACC values among the search results. The values are quantile transformed for easy comparison. The plot suggests no trade-off between the MSE and negative ACC. The optimal configuration in the search results led to the lowest MSE and negative ACC. This aligns with our observation, where negative ACC helped lower MSE in the baseline configuration.

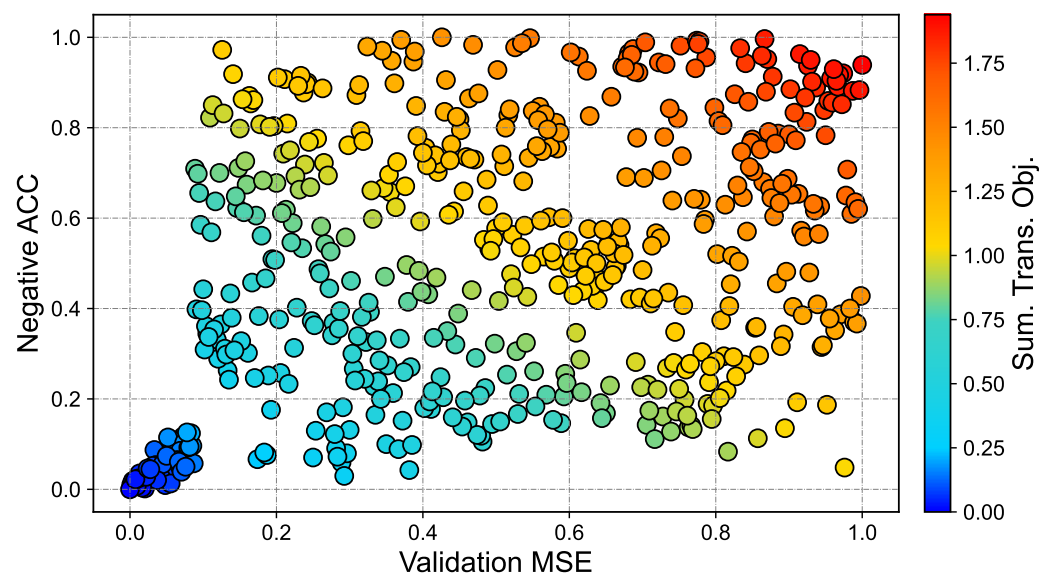


Figure 3. Scatter plot of the quantile-transformed MSE and negative ACC among the search results. The points are color-coded based on the summation of the two objectives, where a lower value indicates better performance.

While the MSE penalizes large errors and focuses on minimizing the average squared error between the forecast and ground truth, ACC pays more attention to the deviation from the mean field, encouraging the model to capture the patterns more accurately. From our experiment, ACC improved the overall learning and helped to reduce the MSE. Therefore, the simple addition of negative ACC might benefit ocean modeling or climate modeling in general using FNOs. We plan to investigate the phenomenon and test this hypothesis in future work.

5.4. Sensitivity Analysis

To understand the impact of the hyperparameters on the FNO performance on the validation set, we conducted a sensitivity analysis using SHAP [51] values. SHAP values quantify the impact of each hyperparameter on the model performance in terms of the validation MSE and ACC. The calculation of SHAP values requires a surrogate model that describes the relationship between the hyperparameters and the objectives. Since the previous search results discussed in Section 5 are based on BO, the samples of hyperparameter configurations are highly correlated. To build the surrogate model using independent samples, we conducted a random search, where the hyperparameter configurations were sampled randomly from the search space. We then used random forest as the surrogate to model the relationship between the hyperparameters and the objectives and calculate the SHAP values. The categorical hyperparameters were encoded ordinal for modeling.

Figure 4 shows the SHAP summary plots of the hyperparameters in the search space with respect to the validation MSE and negative ACC. A lower SHAP value corresponds to the contribution to a lower validation MSE and ACC. For each hyperparameter, its correlation to the objectives is consistent with the previous observations. Meanwhile, among all the hyperparameters, the learning rate, choice of the optimizer, and weight decay magnitude have the most significant impact on the validation MSE. For the validation negative ACC, the number of FNOs, choice of the optimizer, projection layer activation function, and learning rate are the most influential. Notably, the loss weight α controls the importance of the individual term in the loss function, where large values are associated with a lower validation MSE and higher negative ACC. Furthermore, the variation in the number of FNO blocks has a substantial influence on the negative ACC. A higher number of FNO blocks leads to a worse model performance regarding the negative ACC. This could be due to the loss of high frequency modes in the process of stacking more

FNO blocks, which might affect the model’s ability to capture the variations in the data associated with the anomalies. The SHAP values provide a comprehensive understanding of the hyperparameters’ impact on the model performance, which can be used to guide the hyperparameter tuning process, where we can put more emphasis on the most influential hyperparameters.

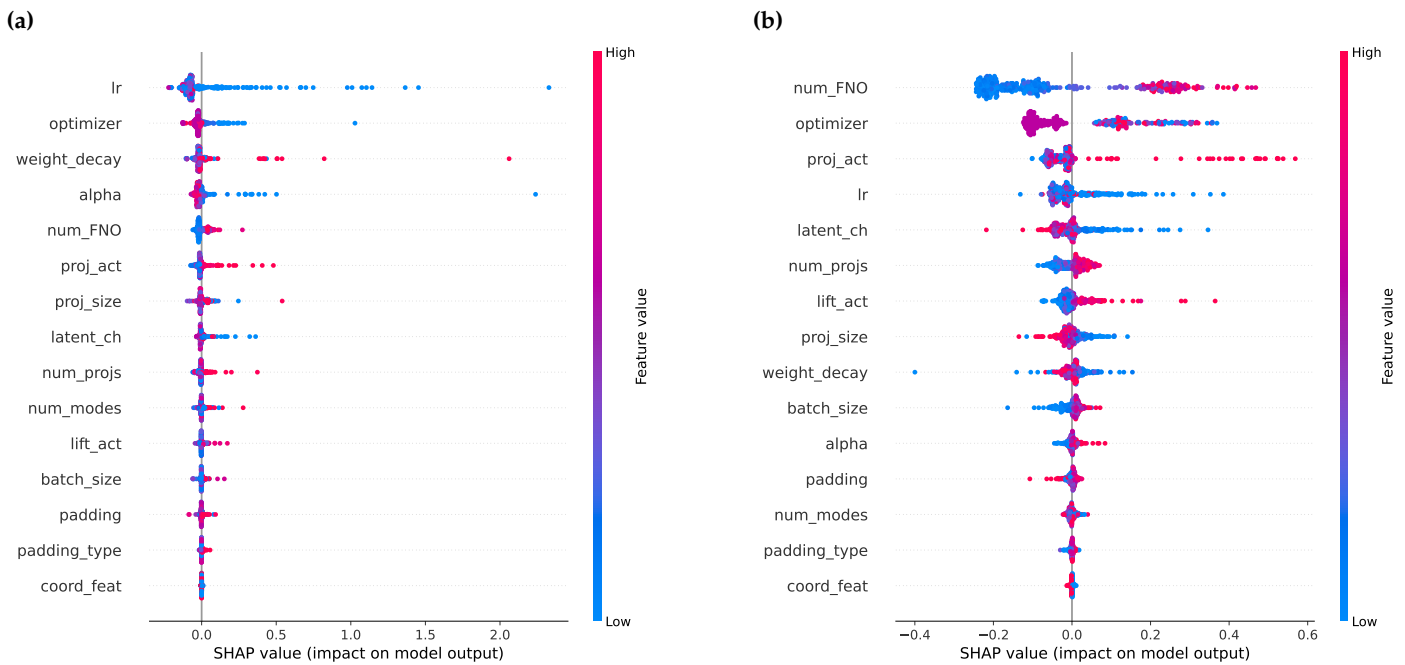


Figure 4. SHAP summary plots of the hyperparameters in the search space with respect to the validation MSE, (a), and negative ACC, (b). A lower SHAP value indicates the hyperparameters’ contribution to decrease the objective values.

5.5. Train with Optimal Configuration

With the search results, we selected the hyperparameter configuration that produced the best performance in both validation MSE and ACC and a model for 100 epochs to report its testing performance and autoregressive rollout scores. Table 3 shows the performance among all the variables of interest using the model trained with the baseline configuration and optimal configuration from the search. The optimal configuration outperforms the baseline for all four variables in both metrics. Figure 5 shows the model predictions on the meridional velocity profiles in the testing set. The model with the optimal configuration produces more accurate forecasts throughout the domain. The optimal configuration has improved the model’s performance, leading to more accurate forecasts.

Table 3. MSE and ACC scores of the baseline model with the optimal configuration from the search results. The downward arrows indicate the lower values are better.

	Baseline		Optimal	
	log(RSE) ↓	log(1 – ACC) ↓	log(RSE) ↓	log(1 – ACC) ↓
Salinity	–2.498	–2.800	–3.143	–3.552
Temperature	–3.061	–3.362	–3.984	–4.286
Meridional V.	–2.067	–2.842	–2.921	–3.254
Zonal V.	–2.303	–2.808	–3.013	–3.349

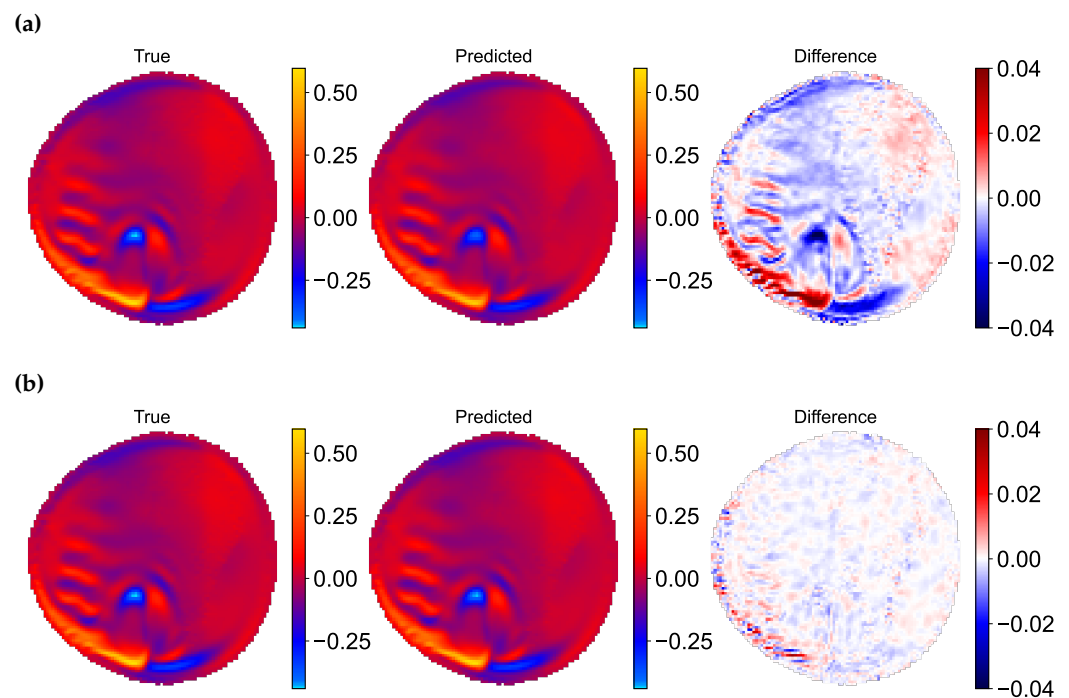


Figure 5. Meridional velocity profiles of model predictions using the testing set. Both models, with different hyperparameters, were trained with 100 epochs using the composite loss function. (a) shows the model performance with the baseline hyperparameter configuration; (b) shows the model with the best configuration from the search results.

The slight improvement is amplified by reapplying the trained models for a longer horizon rollout. Figure 6 shows the model autoregressive rollout performance among the trajectories in the testing dataset. We generated the rollouts by providing the ground truth state variable values at the beginning and used the model to produce the forecast for the next time step. Then, we fed the forecast to the same model as input and obtained the prediction for the next. This process was repeated 29 times, generating forecasting trajectories for an entire month. Both baseline and optimal models start around the same performance. However, the baseline model quickly degrades. Regarding the model with the baseline configurations, the MSE growth rates are similar among the four variables, while meridional velocity shows the earlier ACC degradation and a higher decreasing rate. In comparison, the model using the optimal configurations displays very low error accumulation in MSE and a minimal decrease in the ACC score. Figure 6c shows the visualization of the rollout meridional velocity profiles on Day 10 from the models with baseline and optimal configurations. The optimal configuration has greatly improved the model's autoregressive performance, allowing more accurate longer horizon forecasts. Despite the success, the hyperparameter search process is computationally expensive, and the results are not guaranteed to be the global optimum. More efficient evaluation methods that only require minimal training epochs could be explored to reduce the computational cost.

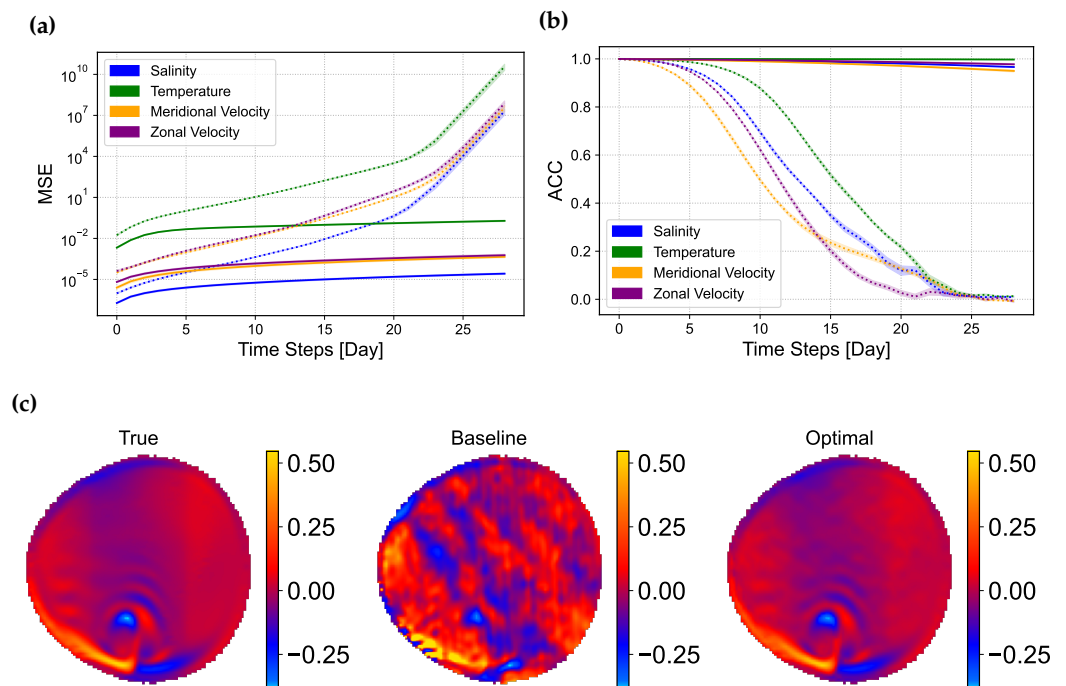


Figure 6. The rollout performance comparison between the models with baseline hyperparameters and searched optimal configurations. (a) shows the rollout MSE values where the error accumulation is much slower for the model with the optimal configuration; (b) shows, compared to the optimal configuration (solid lines), the rollout ACC scores where the model with default configuration (dashed lines) degrades quickly as the rollout horizon increases; (c) shows the model rollout forecasts at Day 10 for meridional velocity profiles.

6. Conclusions

We explored streamlining data-driven ocean modeling using Fourier neural operators with a hyperparameter search, focusing on forecasting four prognostic variables for an idealized baroclinic wind-driven ocean model. We proposed incorporating the negative anomaly correlation coefficient in the loss function to improve model training. The hyperparameter search adopted multiobjective optimization to minimize validation MSE and maximize ACC simultaneously. We utilized parallel coordinate plots and computed SHAP values to study the impact of the hyperparameters on the objectives, through which we identified the correlation between the individual hyperparameter and the objectives as well as the significance among others. The optimal configuration obtained from the search results improves the model performance for all four variables, leading to outstanding improvement in the model's autoregressive rollout performance. The findings suggest hyperparameter optimization can dramatically improve the performance of FNOs by targeting specific objectives, which consolidates high-performing ocean modeling using deep neural networks such as FNOs. Future work includes exploring more efficient function evaluation strategies during the search that allow a greater reduction in resource consumption, such as one-epoch and multi-fidelity approaches, and analyzing the effect of negative ACC on the loss landscape and optimization.

Author Contributions: Conceptualization, Y.S., O.S., R.E. and P.B.; methodology, Y.S., O.S., R.E. and P.B.; data curation, Y.S. and S.H.K.N.; writing—original draft, Y.S.; writing—review & editing, Y.S., O.S., R.E., S.H.K.N., L.V.R. and P.B.; visualization, Y.S.; supervision, P.B.; funding acquisition, S.H.K.N., L.V.R. and P.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research used resources from the Argonne Leadership Computing Facility at Argonne National Laboratory. This material is based upon work supported by the U.S. Department of Energy,

Office of Science, Office of Advanced Scientific Computing Research, and Office of Biological and Environmental Research, Scientific Discovery through Advanced Computing (SciDAC) program under Award Number(s) 9233218CNA000001.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to ongoing research and data curation processes.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kurth, T.; Treichler, S.; Romero, J.; Mudigonda, M.; Luehr, N.; Phillips, E.; Mahesh, A.; Matheson, M.; Deslippe, J.; Fatica, M.; et al. Exascale deep learning for climate analytics. In Proceedings of the SC18: International Conference for High Performance Computing, Networking, Storage and Analysis, Dallas, TX, USA, 11–16 November 2018; pp. 649–660.
2. Rasp, S.; Pritchard, M.S.; Gentile, P. Deep learning to represent subgrid processes in climate models. *Proc. Natl. Acad. Sci. USA* **2018**, *115*, 9684–9689. [[CrossRef](#)] [[PubMed](#)]
3. Nguyen, T.; Brandstetter, J.; Kapoor, A.; Gupta, J.K.; Grover, A. ClimaX: A foundation model for weather and climate. *arXiv* **2023**, arXiv:2301.10343.
4. Gibson, P.B.; Chapman, W.E.; Altinok, A.; Delle Monache, L.; DeFlorio, M.J.; Waliser, D.E. Training machine learning models on climate model output yields skillful interpretable seasonal precipitation forecasts. *Commun. Earth Environ.* **2021**, *2*, 159. [[CrossRef](#)]
5. Pathak, J.; Subramanian, S.; Harrington, P.; Raja, S.; Chattopadhyay, A.; Mardani, M.; Kurth, T.; Hall, D.; Li, Z.; Azizzadenesheli, K.; et al. FourCastNet: A global data-driven high-resolution weather model using adaptive Fourier neural operators. *arXiv* **2022**, arXiv:2202.11214.
6. Cheng, L.; Trenberth, K.E.; Fasullo, J.; Boyer, T.; Abraham, J.; Zhu, J. Improved estimates of ocean heat content from 1960 to 2015. *Sci. Adv.* **2017**, *3*, e1601545. [[CrossRef](#)] [[PubMed](#)]
7. Gou, Y.; Zhang, T.; Liu, J.; Wei, L.; Cui, J.H. DeepOcean: A general deep learning framework for spatio-temporal ocean sensing data prediction. *IEEE Access* **2020**, *8*, 79192–79202. [[CrossRef](#)]
8. Choi, Y.; Park, Y.; Hwang, J.; Jeong, K.; Kim, E. Improving ocean forecasting using deep learning and numerical model integration. *J. Mar. Sci. Eng.* **2022**, *10*, 450. [[CrossRef](#)]
9. Partee, S.; Ellis, M.; Rigazzi, A.; Shao, A.E.; Bachman, S.; Marques, G.; Robbins, B. Using machine learning at scale in numerical simulations with SmartSim: An application to ocean climate modeling. *J. Comput. Sci.* **2022**, *62*, 101707. [[CrossRef](#)]
10. Zhu, Y.; Zhang, R.H.; Moum, J.N.; Wang, F.; Li, X.; Li, D. Physics-informed deep-learning parameterization of ocean vertical mixing improves climate simulations. *Natl. Sci. Rev.* **2022**, *9*, nwac044. [[CrossRef](#)]
11. Guillaumin, A.P.; Zanna, L. Stochastic-deep learning parameterization of ocean momentum forcing. *J. Adv. Model. Earth Syst.* **2021**, *13*, e2021MS002534. [[CrossRef](#)]
12. Zanna, L.; Bolton, T. Data-driven equation discovery of ocean mesoscale closures. *Geophys. Res. Lett.* **2020**, *47*, e2020GL088376. [[CrossRef](#)]
13. Liao, L.; Li, H.; Shang, W.; Ma, L. An empirical study of the impact of hyperparameter tuning and model optimization on the performance properties of deep neural networks. *ACM Trans. Softw. Eng. Methodol. (TOSEM)* **2022**, *31*, 1–40. [[CrossRef](#)]
14. Bi, K.; Xie, L.; Zhang, H.; Chen, X.; Gu, X.; Tian, Q. Pangu-Weather: A 3D High-Resolution Model for Fast and Accurate Global Weather Forecast. *arXiv* **2022**, arXiv:2211.02556.
15. Mustafa, A.; Mikhailiuk, A.; Iliescu, D.A.; Babbar, V.; Mantiuk, R.K. Training a Task-Specific Image Reconstruction Loss. *arXiv* **2021**, arXiv:2103.14616.
16. Zhao, H.; Gallo, O.; Frosio, I.; Kautz, J. Loss functions for image restoration with neural networks. *IEEE Trans. Comput. Imaging* **2016**, *3*, 47–57. [[CrossRef](#)]
17. Johnson, J.; Alahi, A.; Fei-Fei, L. Perceptual losses for real-time style transfer and super-resolution. In Proceedings of the Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016; Proceedings, Part II 14; Springer: Berlin/Heidelberg, Germany, 2016; pp. 694–711.
18. Murphy, A.H.; Epstein, E.S. Skill scores and correlation coefficients in model verification. *Mon. Weather Rev.* **1989**, *117*, 572–582. [[CrossRef](#)]
19. Li, Z.; Kovachki, N.; Azizzadenesheli, K.; Liu, B.; Bhattacharya, K.; Stuart, A.; Anandkumar, A. Fourier neural operator for parametric partial differential equations. *arXiv* **2020**, arXiv:2010.08895.
20. Balaprakash, P.; Salim, M.; Uram, T.D.; Vishwanath, V.; Wild, S.M. DeepHyper: Asynchronous hyperparameter search for deep neural networks. In Proceedings of the 2018 IEEE 25th International Conference on High Performance Computing (HiPC), Bengaluru, India, 17–20 December 2018; pp. 42–51.
21. Balaprakash, P.; Egele, R.; Salim, M.; Maulik, R.; Vishwanath, V.; Wild, S. DeepHyper: A Python Package for Scalable Neural Architecture and Hyperparameter Search. 2018.
22. Lam, R.; Sanchez-Gonzalez, A.; Willson, M.; Wirnsberger, P.; Fortunato, M.; Alet, F.; Ravuri, S.; Ewalds, T.; Eaton-Rosen, Z.; Hu, W.; et al. Learning skillful medium-range global weather forecasting. *Science* **2023**, *382*, 1416–1421. [[CrossRef](#)]

23. Guibas, J.; Mardani, M.; Li, Z.; Tao, A.; Anandkumar, A.; Catanzaro, B. Adaptive Fourier neural operators: Efficient token mixers for transformers. *arXiv* **2021**, arXiv:2111.13587.
24. Li, Z.; Zheng, H.; Kovachki, N.; Jin, D.; Chen, H.; Liu, B.; Azizzadenesheli, K.; Anandkumar, A. Physics-Informed Neural Operator for Learning Partial Differential Equations. *arXiv* **2023**, arXiv:2111.03794.
25. Fanaskov, V.; Oseledets, I. Spectral Neural Operators. *arXiv* **2022**, arXiv:2205.10573.
26. Grady, T.J.; Khan, R.; Louboutin, M.; Yin, Z.; Witte, P.A.; Chandra, R.; Hewett, R.J.; Herrmann, F. Towards Large-Scale Learned Solvers for Parametric PDEs with Model-Parallel Fourier Neural Operators. *arXiv* **2022**, arXiv:abs/2204.01205.
27. Zhang, T.; Trad, D.O.; Innanen, K.A. Learning to solve the elastic wave equation with Fourier neural operators. *Geophysics* **2023**, *88*, T101–T119. [[CrossRef](#)]
28. Bire, S.; Lütjens, B.; Azizzadenesheli, K.; Anandkumar, A.; Hill, C.N. Ocean emulation with Fourier neural operators: Double gyre. *Authorea Prepr.* **2023**.
29. Hutter, F.; Hoos, H.H.; Leyton-Brown, K. Sequential Model-Based Optimization for General Algorithm Configuration. In *Learning and Intelligent Optimization*; Springer: Berlin/Heidelberg, Germany, 2011.
30. Jones, D.R.; Schonlau, M.; Welch, W.J. Efficient global optimization of expensive black-box functions. *J. Glob. Optim.* **1998**, *13*, 455–492. [[CrossRef](#)]
31. Wu, J.; Chen, X.Y.; Zhang, H.; Xiong, L.D.; Lei, H.; Deng, S.H. Hyperparameter optimization for machine learning models based on Bayesian optimization. *J. Electron. Sci. Technol.* **2019**, *17*, 26–40.
32. Egelé, R.; Guyon, I.; Vishwanath, V.; Balaprakash, P. Asynchronous Decentralized Bayesian Optimization for Large Scale Hyperparameter Optimization. In Proceedings of the 2023 IEEE 19th International Conference on e-Science (e-Science), Limassol, Cyprus, 9–13 October 2023; pp. 1–10.
33. Wilson, J.T.; Moriconi, R.; Hutter, F.; Deisenroth, M.P. The reparameterization trick for acquisition functions. *arXiv* **2017**, arXiv:1712.00424.
34. Snoek, J.; Larochelle, H.; Adams, R.P. Practical Bayesian optimization of machine learning algorithms. *Adv. Neural Inf. Process. Syst.* **2012**, *25*.
35. Ginsbourger, D.; Riche, R.L.; Carraro, L. Kriging is well-suited to parallelize optimization. In *Computational Intelligence in Expensive Optimization Problems*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 131–162.
36. Geurts, P.; Ernst, D.; Wehenkel, L. Extremely randomized trees. *Mach. Learn.* **2006**, *63*, 3–42. [[CrossRef](#)]
37. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [[CrossRef](#)]
38. Kadlec, P.; Raida, Z. Multi-objective self-organizing migrating algorithm. *Self-Organizing Migrating Algorithm: Methodology and Implementation*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 83–103.
39. Ehrgott, M. *Multicriteria Optimization*; Springer Science & Business Media: Berlin, Germany, 2005; Volume 491.
40. Égelé, R.; Chang, T.; Sun, Y.; Vishwanath, V.; Balaprakash, P. Parallel Multi-Objective Hyperparameter Optimization with Uniform Normalization and Bounded Objectives. *arXiv* **2023**, arXiv:2309.14936.
41. Radeta, M.; Zuniga, A.; Motlagh, N.H.; Liyanage, M.; Freitas, R.; Youssef, M.A.; Tarkoma, S.; Flores, H.; Nurmi, P. Deep Learning and the Oceans. *Computer* **2022**, *55*, 39–50. [[CrossRef](#)]
42. Er, M.J.; Chen, J.; Zhang, Y.; Gao, W. Research Challenges, Recent Advances, and Popular Datasets in Deep Learning-Based Underwater Marine Object Detection: A Review. *Sensors* **2023**, *23*, 1990. [[CrossRef](#)] [[PubMed](#)]
43. Zrira, N.; Kamal-Idrissi, A.; Farssi, R.; Khan, H.A. Time series prediction of sea surface temperature based on BiLSTM model with attention mechanism. *J. Sea Res.* **2024**, *198*, 102472. [[CrossRef](#)]
44. Li, X.; Liu, B.; Zheng, G.; Ren, Y.; Zhang, S.; Liu, Y.; Gao, L.; Liu, Y.; Zhang, B.; Wang, F. Deep-learning-based information mining from ocean remote-sensing imagery. *Natl. Sci. Rev.* **2020**, *7*, 1584–1605. [[CrossRef](#)] [[PubMed](#)]
45. Colin, A.; Tandeo, P.; Peureux, C.; Husson, R.; Longépé, N.; Fablet, R. Rain regime segmentation of Sentinel-1 observation learning from NEXRAD collocations with Convolution Neural Networks. *IEEE Trans. Geosci. Remote Sens.* **2024**, *62*, 4202914. [[CrossRef](#)]
46. Sun, Y.; Cucuzzella, E.; Brus, S.; Narayanan, S.H.K.; Nadiga, B.; Van Roekel, L.; Hüchelheim, J.; Madireddy, S. Surrogate Neural Networks to Estimate Parametric Sensitivity of Ocean Models. *arXiv* **2023**, arXiv:2311.08421.
47. Wolfram, P.J.; Ringler, T.D.; Maltrud, M.E.; Jacobsen, D.W.; Petersen, M.R. Diagnosing isopycnal diffusivity in an eddying, idealized midlatitude ocean basin via Lagrangian, in situ, global, high-performance particle tracking (LIGHT). *J. Phys. Oceanogr.* **2015**, *45*, 2114–2133. [[CrossRef](#)]
48. Alrasheedi, F.; Zhong, X.; Huang, P.C. Padding Module: Learning the Padding in Deep Neural Networks. *IEEE Access* **2023**, *11*, 7348–7357. [[CrossRef](#)]
49. Bengio, Y. Practical recommendations for gradient-based training of deep architectures. *arXiv* **2012**, arXiv:1206.5533.
50. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2017**, arXiv:1412.6980.
51. Lundberg, S.; Lee, S.I. A Unified Approach to Interpreting Model Predictions. *arXiv* **2017**, arXiv:1705.07874.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.