Ocean Emulation with Fourier Neural Operators: Double Gyre

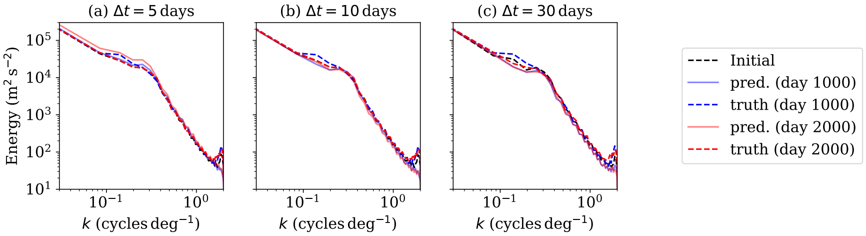
Suyash Bire[1], Björn Lütjens[1], Kamyar Azizzadenesheli[2], Anima Anandkumar[3], and Christopher N. Hill[1]

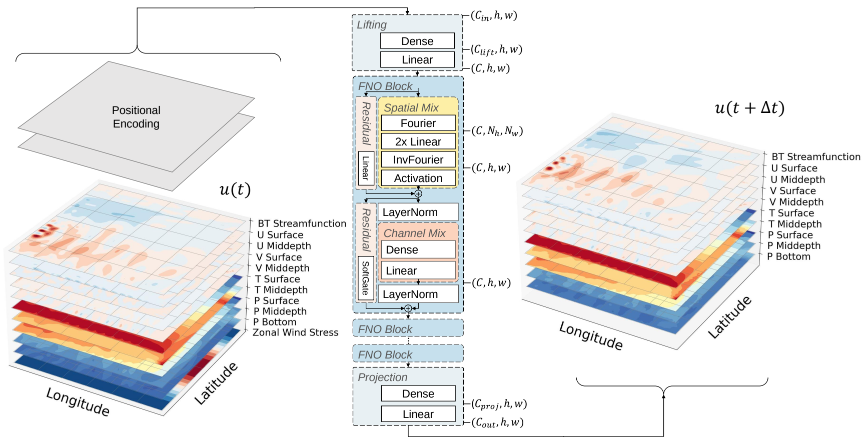
[1]Massachusetts Institute of Technology
[2]Nvidia
[3]California Institute of Technology

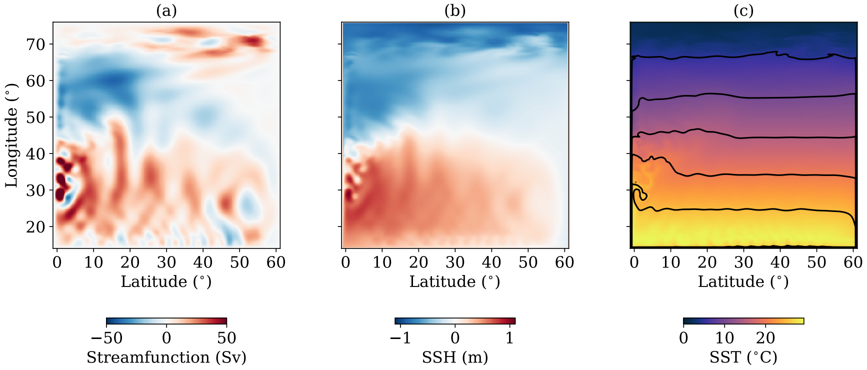
November 27, 2023

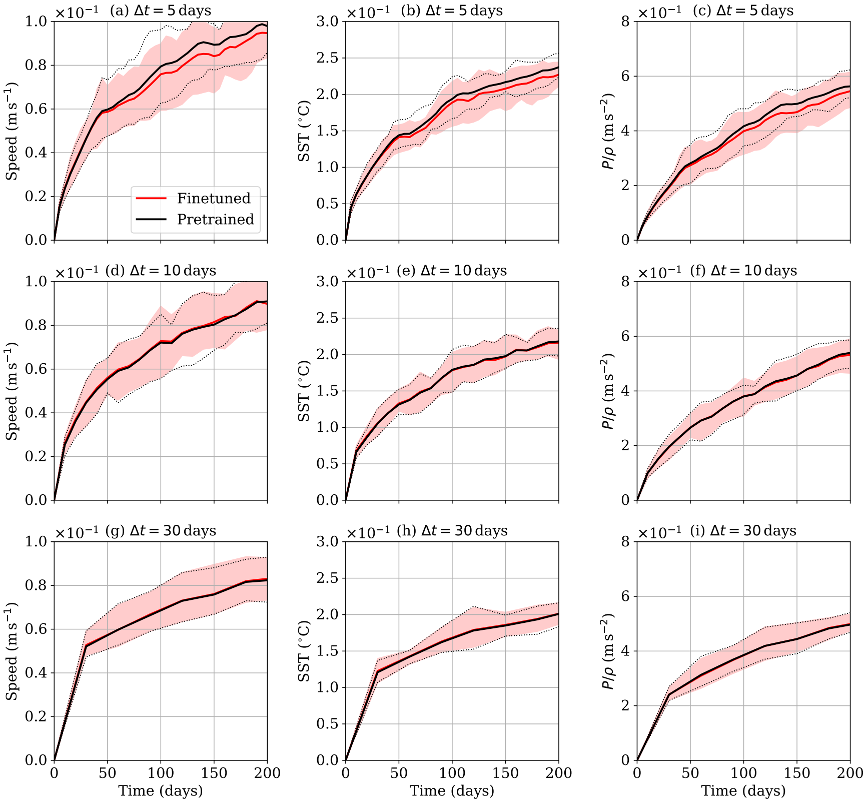
Abstract

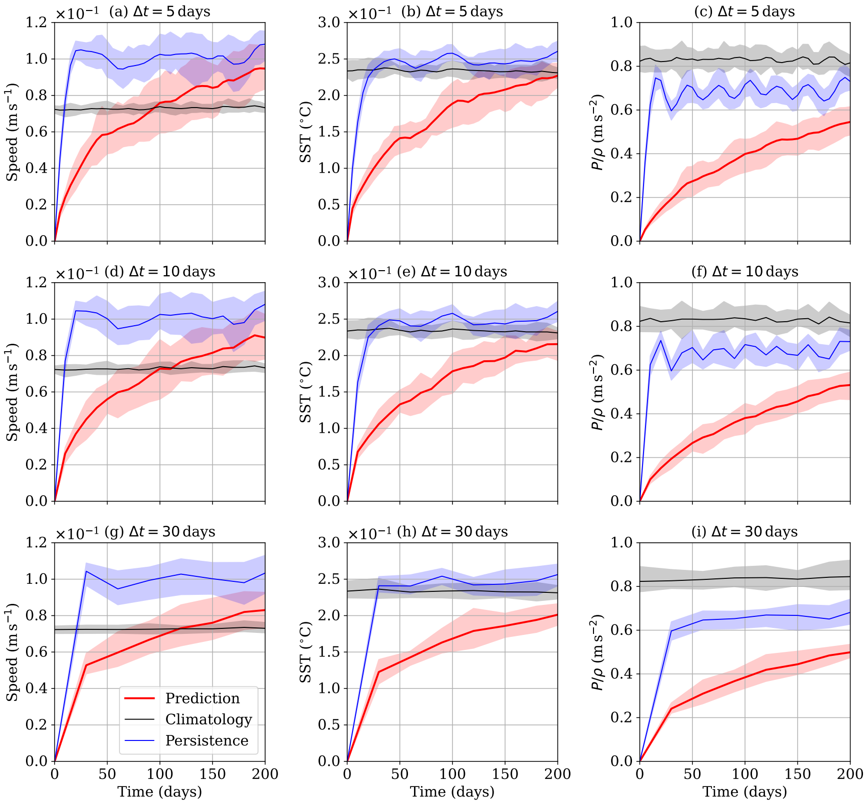
A data-driven emulator for the baroclinic double gyre ocean simulation is presented in this study. Traditional numerical simulations using partial differential equations (PDEs) often require substantial computational resources, hindering real-time applications and inhibiting model scalability. This study presents a novel approach employing neural operators to address these challenges in an idealized double-gyre ocean simulation. We propose a deep learning approach capable of learning the underlying dynamics of the ocean system, complementing the classical methods, and effectively replacing the need for explicit PDE solvers at inference time. By leveraging neural operators, we efficiently integrate the governing equations, providing a data-driven and computationally efficient framework for simulating the double-gyre ocean circulation. Our approach demonstrates promising results in terms of accuracy and computational efficiency, showcasing the potential for advancing ocean modeling through the fusion of neural operators and traditional oceanographic methodologies. In comparison to a dynamical numerical model, we obtain 600x speedups allowing us to create 2000-day ensembles in tens of seconds instead of hours.

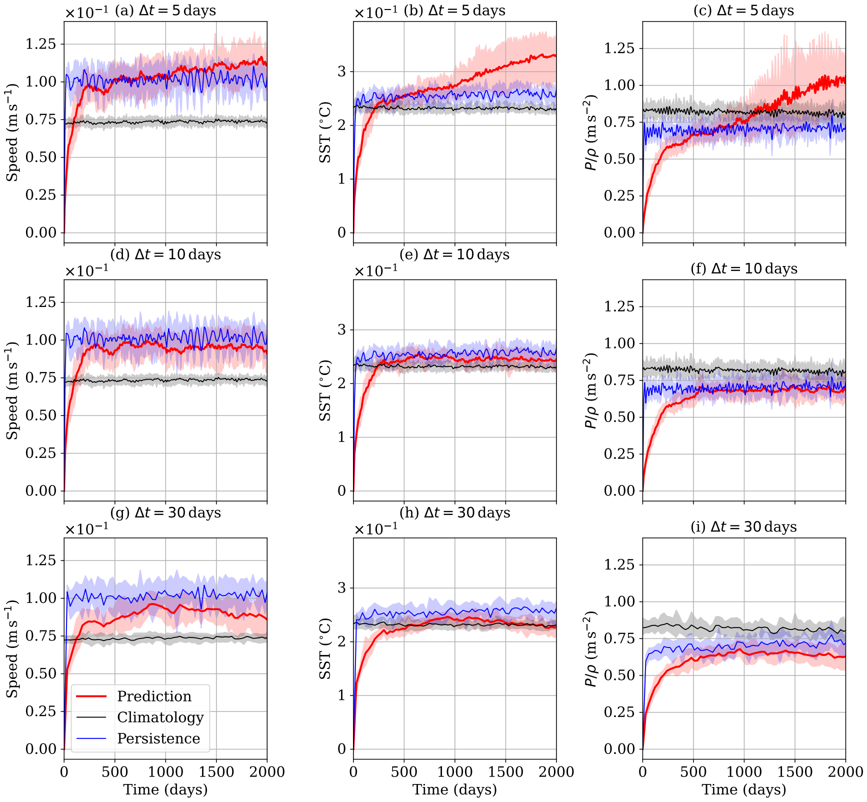



Positional Encoding
$u(t)$
BT Streamfunction
U Surface
U Middepth
V Surface
V Middepth
T Surface
T Middepth
P Surface
P Middepth
P Bottom
Zonal Wind Stress
Longitude
Latitude
Lifting
Dense
Linear
(C_{in}, h, w)
(C_{lift}, h, w)
(C, h, w)
FNO Block
Residual
Linear
Spatial Mix
Fourier
2x Linear
InvFourier
Activation
(C, N_h, N_w)
(C, h, w)
LayerNorm
Residual
SoftGate
Channel Mix
Dense
Linear
LayerNorm
(C, h, w)
FNO Block
FNO Block
Projection
Dense
Linear
(C_{proj}, h, w)
(C_{out}, h, w)
$u(t + \Delta t)$
BT Streamfunction
U Surface
U Middepth
V Surface
V Middepth
T Surface
T Middepth
P Surface
P Middepth
P Bottom
Longitude
Latitude


(a)
(b)
(c)
Longitude (°)
70
60
50
40
30
20
0
10
20
30
40
50
60
Latitude (°)
−50
0
50
Streamfunction (Sv)
−1
0
1
SSH (m)
0
10
20
SST (°C)


(a) $\Delta t = 5$ days
(b) $\Delta t = 5$ days
(c) $\Delta t = 5$ days
(d) $\Delta t = 10$ days
(e) $\Delta t = 10$ days
(f) $\Delta t = 10$ days
(g) $\Delta t = 30$ days
(h) $\Delta t = 30$ days
(i) $\Delta t = 30$ days
$\times 10^{-1}$
Speed (m s^{-1})
SST (°C)
P/ρ (m s^{-2})
Time (days)
Finetuned
Pretrained
0.0
0.2
0.4
0.6
0.8
1.0
0.5
1.5
2.0
2.5
3.0
0
2
4
6
8
50
100
150
200


$\times10^{-1}$ (a) $\Delta t = 5$ days
$\times10^{-1}$ (b) $\Delta t = 5$ days
(c) $\Delta t = 5$ days
$\times10^{-1}$ (d) $\Delta t = 10$ days
$\times10^{-1}$ (e) $\Delta t = 10$ days
(f) $\Delta t = 10$ days
$\times10^{-1}$ (g) $\Delta t = 30$ days
$\times10^{-1}$ (h) $\Delta t = 30$ days
(i) $\Delta t = 30$ days
Speed ($\mathrm{m\,s^{-1}}$)
SST (°C)
P/ρ ($\mathrm{m\,s^{-2}}$)
Time (days)
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0.5
1.5
2.0
2.5
3.0
0
50
100
150
200
Prediction
Climatology
Persistence


(a) $\Delta t = 5$ days
(b) $\Delta t = 5$ days
(c) $\Delta t = 5$ days
(d) $\Delta t = 10$ days
(e) $\Delta t = 10$ days
(f) $\Delta t = 10$ days
(g) $\Delta t = 30$ days
(h) $\Delta t = 30$ days
(i) $\Delta t = 30$ days
$\times 10^{-1}$
Speed (m s^{-1})
SST (°C)
P/ρ (m s^{-2})
0.00
0.25
0.50
0.75
1.00
1.25
0
1
2
3
0
500
1000
1500
2000
Time (days)
Prediction
Climatology
Persistence

Ocean Emulation with Fourier Neural Operators: Double Gyre

Suyash Bire[1], Björn Lütjens[1], Kamyar Azizzadenesheli[2], Animashree Anandkumar[2,3], Chris Hill[1]

[1]Earth, Atmospheric, and Planetary Sciences, Massachusetts Institute of Technology, Cambridge, MA 02139 US
[2]NVIDIA Corp., Santa Clara, Santa Clara, CA 95051, US
[3]California Institute of Technology, Pasadena, , CA 91125, US

Key Points:

- We present an emulator of a simplified ocean simulation called the double gyre.
- The emulator is based on Fourier neural operators.
- The emulator is capable of producing long ensembles which is a major challenge for data-driven methods.

Corresponding author: Suyash Bire, `bire@mit.edu`

Abstract

A data-driven emulator for the baroclinic double gyre ocean simulation is presented in this study. Traditional numerical simulations using partial differential equations (PDEs) often require substantial computational resources, hindering real-time applications and inhibiting model scalability. This study presents a novel approach employing neural operators to address these challenges in an idealized double-gyre ocean simulation. We propose a deep learning approach capable of learning the underlying dynamics of the ocean system, complementing the classical methods, and effectively replacing the need for explicit PDE solvers at inference time. By leveraging neural operators, we efficiently integrate the governing equations, providing a data-driven and computationally efficient framework for simulating the double-gyre ocean circulation. Our approach demonstrates promising results in terms of accuracy and computational efficiency, showcasing the potential for advancing ocean modeling through the fusion of neural operators and traditional oceanographic methodologies. In comparison to a dynamical numerical model, we obtain 600x speedups allowing us to create 2000-day ensembles in tens of seconds instead of hours.

Plain Language Summary

We propose learning the dynamics of a simplified ocean simulation using a data-driven architecture called neural operators. Neural operators, recognized for their suitability in scientific computing and ability to learn mappings between function spaces, offer fast, differentiable surrogate models. This approach demonstrates the possibility of rapid modeling of the realistic ocean in the future.

1 Introduction

Fourier neural operators (FNO) have gained popularity in modeling of various physical phenomena that are governed by partial differential equations (Li et al., 2020b, 2020a). They have been shown to successfully emulate fluid flow problems (Wen et al., 2021), shallow water equation solvers, and numerical weather prediction models (Pathak et al., 2022; Kurth et al., 2022) among others. The strength of FNOs lies in their ability to learn mappings between continuous function spaces from data (Kovachki et al., 2021). For example, when employed in weather prediction, FNO can learn the relationships between two states of the atmosphere separated by a given prediction interval, effectively emulating the time-stepping process. Kurth et al. (2022) show that their model, FourCastNet, which utilizes Fourier neural operators is able to match the medium-range prediction accuracy of ECMWF's (European Centre for Medium-Range Weather Forecasts) dynamical numerical weather prediction model at a fraction of the computational cost and time (80000 times faster). However, their model when iteratively timestepped cannot remain stable for longer than 25 days. This limitation is attributed to the problems arising due to the convergence of meridians at the poles. This issue is alleviated by the spherical Fourier neural operator (SFNO) of Bonev et al. (2023) by employing the more general spherical harmonic transform in the meridional direction. SFNO is able to generate ensembles of upto a year as compared to 25 days of FourCastNet.

Since FNOs work so well for atmospheric emulation a natural progression is to extend them to emulate ocean simulations. In this study, as an initial foray into ocean emulation, we focus our attention on a simplified simulation known as the "Double Gyre" (Bryan, 1963; Cox & Bryan, 1984). The double gyre simulation is an idealized version of a northern hemispheric ocean basin forced by the easterly-westerly wind structure and equator-to-pole sea surface temperature gradient. Although SFNO produces better results on the sphere than FNO our setup does not have poles, therefore we employ an FNO. We use training and testing data generated by running three MITgcm (Marshall et al.,

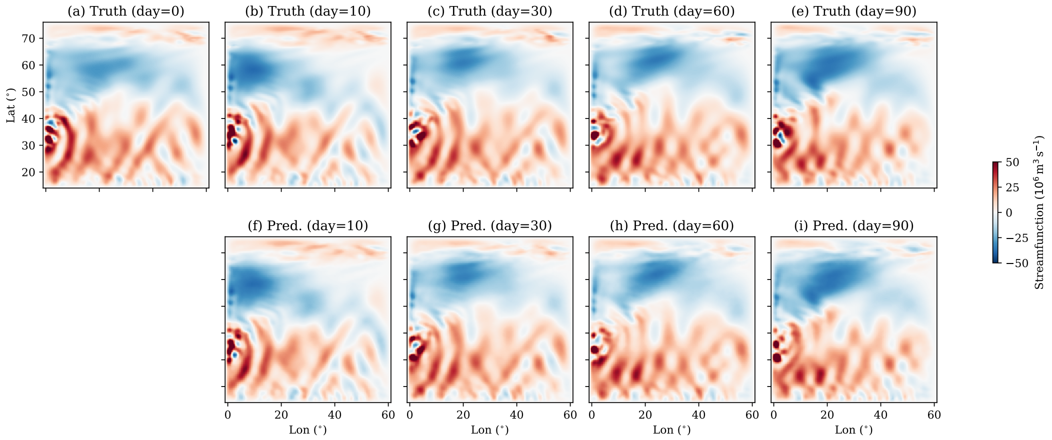


Figure 1. Top row shows the barotropic streamfunction from the MITgcm simulation, while bottom shows that from the FNO prediction.

1997) simulations and compare the accuracy of our emulator with MITgcm. We show that the FNO-based emulator is up to 1000 times faster than MITgcm, and can produce long ensembles ranging into thousands of days.

A comparison of vertically integrated (barotropic) streamfunction from the MITgcm simulation (ground truth) with that from the FNO prediction is shown in fig. 1. Positive value in the southern half of the domain represents the subtropical gyre where the mean flow is clockwise, while negative value represents the subpolar gyre with anticlockwise flow. Our double gyre simulation has waves propagating westward and geostrophic turbulence near the western boundary. The predictions in the bottom panels are generated iteratively, that is, the first prediction in panel f is made from the initial condition in panel a but all the other predictions are made from the previous prediction. The prediction interval is 10 days. The excellent match between the predictions in the bottom panels and truth in the top panels shows that the FNO emulator is good at learning the dynamics of these waves and the western boundary turbulence. At around the 90th day the prediction starts to diverge significantly from the ground truth as the underlying simulation becomes inherently unpredictable at this timescale.

Almost all of the computational cost is incurred during the training phase. For example, the emulator in its present form requires 4 hours for pretraining and subsequently 2 hours for multistep finetuning (see 3.3.4) on an Nvidia Tesla V-100 GPU. The subsequent cost of producing a single ensemble is at least two orders of magnitude smaller than that of a traditional numerical model. For example, the 20-year long control MITgcm simulation was produced on 4 Intel E5 Hasswel-EP processors with 16 cores each running at 2.1 GHz in wall-clock time of 6.5 hours. In comparison, 20-year ensemble from FNO can be produced in 130 s, 60 s, or 40 s for prediction interval (Δt) of 5, 10, and 30 days, respectively, on one Nvidia Tesla V-100 GPU. This gives us speedups of 200, 400, and 600 times, respectively, for Δt of 5, 10, and 30 days. Note that this is not an apples-to-apples comparison because the two models were tested on different systems. However if they were tested on the same machine we are confident that the emulator would be orders of magnitude faster. Since the numerical model employs partial differential equation solvers, it has to use a timestep of 300 s in accordance with the Courant-Freidrich-Lewy condition. Our emulator has no such constraints as long as the prediction interval is within the decorrelation timescale.

Rest of the paper is organized as follows. Section 2 discusses prior studies employing data-driven methods in weather and climate science. In section 3, we describe the numerical model setup as well as the emulator. Section 4 describes the solution obtained in the numerical model, while section 5 compares the predictions from our emulator with the numerical model. Discussion and summary is in section 6.

2 Related Work

Numerical weather and climate prediction models comprise finite difference equations written on discrete grids. The finer the grid, the more computationally expensive the model becomes. There have been significant improvements in the forecast skill of climate models in recent decades, but the computational requirements have only increased even with more efficient computers (Bauer et al., 2015). In search of computational efficiency (e.g. Dewitte et al., 2021) as well as in light of increasing complexity of parameterization schemes (e.g. Christensen & Zanna, 2022), data-driven approaches have recently become attractive. The appeal of data-driven approaches, especially deep learning, lies in their ability to learn non-linear relationships hidden in large datasets at relatively low computational costs.

One approach to improve efficiency is to decrease the resolution of the models. However, decreasing the resolution requires good parameterization schemes of processes occurring on scales smaller than the grid size (Christensen & Zanna, 2022). To be clear, parameterization schemes are also required in high-resolution models, because there are always processes at even smaller scales, but this problem becomes especially pronounced in coarse-resolution models. A number of studies have recently shown that it is possible to reasonably represent the effect of small-scale processes in terms of resolved processes using machine learning methods (e.g Bolton & Zanna, 2019; Zanna & Bolton, 2020; Guillaumin & Zanna, 2021; Yuval & O'Gorman, 2023). Another approach is to leverage machine learning approaches in downscaling (super-resolving) numerical predictions made using relatively cheap coarse resolution models (e.g. Höhlein et al., 2020; Jiang et al., 2023).

Recently, yet another approach that has become popular involves predicting a future state from an earlier state. In essence, this approach attempts to replace the entire numerical model with a machine-learning emulator. The emulator makes sense in certain scenarios. For example, in weather prediction operationally, the exact same numerical model is run every few hours, just with new parameters. Emulators can potentially make this process more efficient. So far, this approach has only been applied to atmospheric processes like weather forecasts, precipitation predictions, etc. For example, Dueben and Bauer (2018) use neural networks and training data from ERA5 reanalysis (Hersbach et al., 2020) to predict increments in 500 hpa geopotential heights one hour apart. They employ two fully-connected neural networks, first a local network that predicts the state at a particular location from only its neighbors, and a global network that predicts the state at a given location from all other locations. They find that their local network performs better than the global one. They chain successive forecasts together and argue that neural networks would be limited to making predictions of very short timescales (a few hours). This shortcoming is attributed to the spatially limited nature of their neural network, as long timescale predictions would require the model to learn relationships across longer length scales.

Scher (2018) overcome this limitation by using deep convolutional neural networks and chaining successive forecasts together to form long predictions. They use numerical climate simulations with simplified physics as ground truth and draw training and testing samples from them. Their neural network predicts horizontal velocities, temperatures, and geopotential heights at 10 levels each. They train their emulator to operate over intervals of 1 to 14 days. They employ a neural network with the encoder-decoder structure, which consists of layers where the dimensionality of the input is first reduced and then increased back to the original dimensionality. This allows the output of the network to be of the same dimension as the input. A suitably defined cost function, like mean square error with respect to a later time step, allows the output of the network to be interpreted as a dynamical state at this later time. Further, they show that output from the neural network can be fed back into it and iteratively chained to create long ensembles of simulations. They find that networks that directly predict future states at intermediate intervals, around 5 days, perform better than iterative 1-day forecasts chained 5 times. They further extend their study in Scher and Messori (2019) with a hierarchy of more complicated models to generate training data and show that neural networks developed for simple models show promise in emulating more complex numerical models. They also find that long climate emulations fail to capture features like the location of storm tracks or seasonal variations accurately.

Rasp et al. (2020) provide a database, WeatherBench, which consists of standardized data necessary for good medium-range forecast. They, and more recently Ben-Bouallegue et al. (2023), also suggest metrics, like root mean square error (RMSE) and anomaly correlation coefficient (ACC) of specific fields, for benchmarking data-driven methods with state-of-the-art numerical weather prediction models. Rasp and Thuerey (2021) leverage this dataset and demonstrate the use of ResNET, a convolutional neural network with residual connections and encoder-decoder layout, to perform short-range weather prediction. Further, Weyn et al. (2019) and Weyn et al. (2020) apply an improved convolutional neural network with deeper residual connections, U-Net, and a recurrent neural network with long/short-term-memory (LSTM) to perform similar predictions. The introduction of residual connections is a major improvement over previous studies. Residual connections allow the networks to efficiently learn autocorrelations that are ubiquitous in weather data. Another improvement introduced by Weyn et al. (2019) is using two successive timesteps as inputs to predict two successive future timesteps. This presumably allows their model to learn conservation principles and makes their long-term climatology runs more reasonable. In spite of these improvements, the forecast skill of all of these models is inadequate compared to dynamical numerical weather prediction models.

However, recently, the incorporation of ideas from graph neural networks and transformer architectures has significantly improved the forecast skill of data-driven prediction models. The ability of these architectures to learn relationships between random locations seems to be particularly amenable to emulating fluid flows. For example, Cachay et al. (2021) and Zhou and Zhang (2023) predict ENSO up to a few months in advance using graph neural networks and geo-transformer, respectively. Some other recent models that accurately predict short to medium-term weather are also based on ideas from graph neural networks and transformers (Bi et al., 2022; Lam et al., 2022; Nguyen et al., 2023).

In spite of these improvements, all the above-mentioned methods only apply on fixed grids of their respective training data, do not have implicit knowledge that the predicted variables are functions on the sphere, and can not be evaluated at locations off their respective grids. These limitations are considerable challenges to making these methods more generally applicable. FourCastnet and SFNO address some of these underlying issues. An end to end neural operator approach of SFNO of Bonev et al. (2023) improves the skill of FourcastNet by decomposing the meridional direction in spherical harmonics rather than Fourier harmonics. This allows them to address the singularity at the poles, which Fourier decomposition on a rectangular domain cannot address. Furthermore, this model enables training and evaluation on various grids and datasets and allows evaluation of the atmosphere's state at any point on Earth, a property lacking in prior neural network-based approaches. At the time of writing ECMWF has operationalized (*ECMWF*

Charts, n.d.) FourCastNet, Graphcast (Lam et al., 2022), and Pangu-Weather (Bi et al., 2022).

3 Methods

3.1 Numerical simulations

We generate a ground-truth dataset by simulating a double-gyre (Bryan, 1963; Berloff & Meacham, 1998), an idealized representation of a real northern hemispheric ocean basin, using MITgcm, a dynamical model for ocean simulation (Marshall et al., 1997). The domain extends from 0° to 62° in the east-west direction and from $y_\mathrm{s} = 10°\mathrm{N}$ to $y_\mathrm{n} = 72°\mathrm{N}$ in the north-south direction. The resolution is $0.25° \times 0.25°$ implying 248×248 grid points on this area on the sphere. In the vertical direction we employ 15 levels spanning a depth of 2000 m with the grid thickness increasing from 50 m at the surface to 190 m at the bottom. This setup, a standard setup in MITgcm, is similar to the simulation in Cox and Bryan (1984).

The model is forced by zonal wind at the surface given by

$$\tau^x = -\tau_0 \cos\left(2\pi \frac{y - y_\mathrm{s}}{y_\mathrm{n} - y_\mathrm{s}}\right), \tag{1}$$

where τ_0 is the maximum wind velocity which can be specified. This profile of zonal wind forcing is motivated by the realistic meridional profile of wind, that is, westerlies at mid-latitudes and easterlies in the tropics. Additionally, temperature at the surface is relaxed to a meridional profile given by

$$T_\mathrm{surf} = \frac{T_\mathrm{max} - T_\mathrm{min}}{y_\mathrm{s} - y_\mathrm{n}} * (y_\mathrm{n} - y) + T_\mathrm{min}, \tag{2}$$

where T_min and T_max are the temperatures at the northern and southern edges of the domain, respectively, and can be specified. In this study, we have maintained $T_\mathrm{max} = 30\,°\mathrm{C}$ and $T_\mathrm{min} = 10\,°\mathrm{C}$, which represent a realistic equator-to-pole SST gradient. The meridional variation of SST induces a three-dimensional overturning circulation in the basin. The horizonal viscosity and diffusivity both are set to $500\,\mathrm{m}^2\,\mathrm{s}^{-1}$, while the vertical viscosity is $10^{-2}\,\mathrm{m}^2\,\mathrm{s}^{-1}$ and vertical diffusivity is $10^{-5}\,\mathrm{m}^2\,\mathrm{s}^{-1}$. We arrive at these values through trial and error. These values allow us to minimize numerical noise in the finite-difference schemes while retaining physical turbulence in the circulation patterns. No normal flow and no-slip boundary conditions are applied at all the lateral boundaries. Bottom boundary condition is free slip.

We perform three simulations by varying the wind forcing as shown in the table 1. Each simulation is integrated to statistically steady state after which velocities, pressure, and temperature are saved at an interval of 1 day for 200 years. Overall, 7200 snapshots from each simulation are saved.

Table 1. Numerical simulations

ID	Experiment	τ_0 ($\mathrm{N\,m^{-2}}$)	Split
1	Control	0.1	Train
2	Low Wind	0.075	Train, Val
3	High Wind	0.125	Train

3.2 Training and validation datasets

Data from simulations 2 and 3 (table 1) are entirely allocated for training, while that from simulation 1 is split evenly. First half is appended to the training dataset while the second half is allocated for validation. Thus, we make sure that the validation dataset is unseen during the training phase. This arrangement, a standard practice in machine learning studies, allows us to ensure that the neural operator does not overfit our training data. We do not use a third held-out test dataset.

More specifically, similar to Scher and Messori (2019) and Pathak et al. (2022), in this work, two-dimensional fields of various variables are passed as channels to the deep learning model. The first two channels are zonal velocities at surface and middepth, the second and third channels are for meridional velocities again at the surface and middepth. The next two channels are for surface and middepth temperature, while the next three channels are for pressure at the surface, middepth, and bottom. Due to the imposed SST relaxation, the vertical structure of the flow in the numerical model is baroclinic, that is there is a net flow towards the north of the basin near the surface and a return flow towards the south at mid-depth levels. Therefore, we determine that a good representative sample of the flow would consist of surface layers as well as mid-depth layers. The pressure at surface also serves as direct proxy for sea surface height, which is a barotropic (vertically integrated) quantity. We also add a channel representing vertically integrated streamfunction as an additional barotropic quantity. Thus, we have 9 channels/variables representing the state of the ocean as input to the neural operator. Vertical structure in the ocean does not change drastically below the thermocline, therefore we deem that these channels are sufficient. Our testing suggests that adding more channels representing intermediate depths only increases the model size without much accuracy gain.

Additionally, we also append a two-dimensional map of wind forcing obtained according to eq. (1). This allows the neural operator to know when the intensity of the wind is changed. In the future, we plan to integrate this ocean model with an atmospheric simulation. Wind and surface heat and moisture fluxes are the primary avenues through which the atmosphere and ocean influence each other. For the current simulations, we only include wind as a forcing parameter. Thus, dataset for each simulation consists of 11 channels as shown in fig 2.

3.3 Fourier Neural Operator Architecture

Fourier neural operator architecture employed here takes 2-dimensional fields of variables and parameters at any given instant as input and gives the same fields at a later instant similar to the encoder-decoder structure employed by Scher (2018) and Scher and Messori (2019). The fields can be thought of as functions defined on a 2-dimensional domain. Each of these 2-dimensional fields are stacked together to form a third dimension called channels or co-dimensions. In our case, the number of co-dimensions of the input function, u, is 11, and given the spatial resolution of $h{\times}w$, h being the number of latitudes and w being the number of longitudes, the input function in the form of a tensor is of size $11\times h\times w$ as explained in section 3.2. The values of parameters used are shown in table 2. A graphical representation of the architecture is shown in fig. 2. The input tensor is shown on the left in fig. 2.

3.3.1 Positional encoding and lifting

In the first stage, called the positional encoding, we add two co-dimensions that represent the relative distance between each grid point in latitude and longitude directions using sines and cosines similar to Vaswani et al. (2017). The role of these layers is to provide the neural operator with a sense of location. Thus, the function now attains a shape $C_{\text{in}} \times h \times w$. In the study of Pathak et al. (2022), a patching operation

Table 2. FNO hyperparameters

Hyperparameter	Value
Batch size	8
Learning rate	0.01
h, w	(248,248)
$N_\mathrm{h}, N_\mathrm{w}$	(64, 64)
L	3
C_in, C_out	13, 10
C_lift, C_proj	256, 256
C	128
C_hidden	$4 \times C$
σ	ReLU
Δt	5, 10, and 30 days

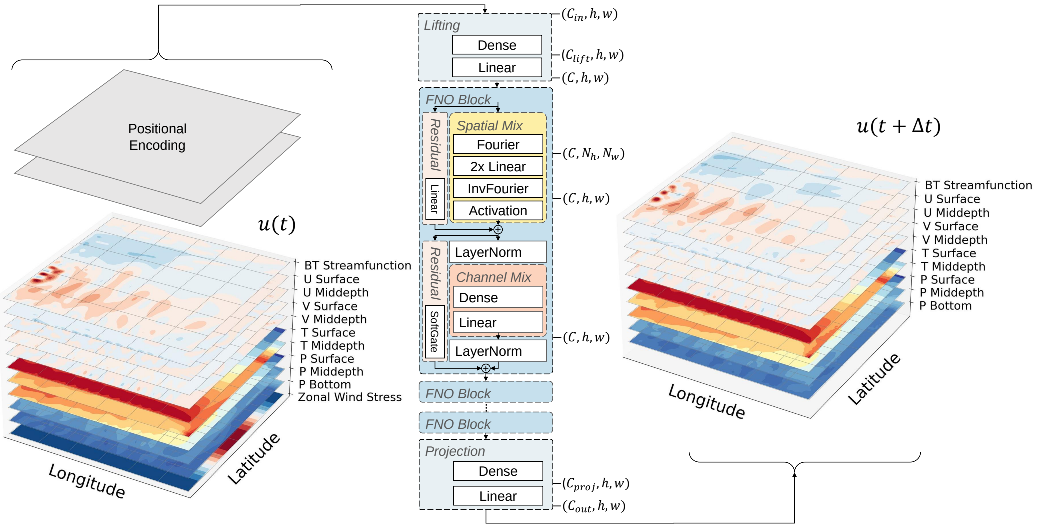


Figure 2. A rough sketch of the FNO architecture is shown. The input function, u, has 11 channels representing velocities, temperature, pressure, streamfunction, and wind. Two channels encoding the latitude-longitude positions are added. The output function has 10 channels representing velocities, temperature, pressure, and streamfunction. The operator is composed of the lifting, FNO, and projection blocks. We choose three FNO blocks as shown.

is also applied. While this allows reducing the size of the input array in the horizontal direction, it comes at the cost of losing the resolution-invariance property of the FNO architecture. Therefore, we avoid the patching operation in this study.

The channel dimension consisting of C_{in} channels is further expanded into C channels by passing the tensor to a two layer fully-connected neural network with learnable weights:

$$\begin{aligned} u &\leftarrow \sigma(W_{\text{lift},1}\, u + b_{\text{lift},1}), && (3)\\ u &\leftarrow W_{\text{lift},2}\, u + b_{\text{lift},2}, && (4) \end{aligned}$$

where $W_{\text{lift},1}$ and $W_{\text{lift},2}$ have shapes $C_{\text{lift}}\times C_{\text{in}}$ and $C\times C_{\text{lift}}$, respectively. Biases $b_{\text{lift},1}$ and $b_{\text{lift},2}$ have shapes C_{lift} and C, respectively. At this stage the input function of C_{in} co-dimensions is encoded into a latent space of C co-dimensions.

3.3.2 FNO Block: Spatial and channel mixing

In the second stage, the encoded tensor of shape $C\times h\times w$ is passed into the FNO blocks (Li et al., 2020a, 2020b). FNO blocks consist of spatial mixing and channel mixing steps. First, the input tensor u is recast into the shape $h\times w\times C$. Then a Fourier transform is applied along horizontal dimensions and only the first N_{h} and N_{w} modes are retained so that the shape of X becomes $N_{\text{h}}\times N_{\text{w}}\times C$. The phases and amplitudes associated with each wavenumber are then transformed by multiplying with block-specific learnable weights R_1^l, R_2^l, R_3^l, and R_4^l of shapes $N_{\text{h}}\times N_{\text{h}}$. An inverse transform is then taken which transforms the tensor back into latitude-longitude space. The tensor is then recast into the shape $C \times h \times w$. This process is called spatial mixing (Lee-Thorp et al., 2021; Rao et al., 2021; Guibas et al., 2021) because it allows the model to learn relationships in the horizontal (latitude-longitude) dimensions. Spatial mixing operation can be mathematically represented as

$$\begin{aligned} u_{\text{orig}} &\leftarrow u, && (5)\\ u &\leftarrow F(u), && (6)\\ u &\leftarrow \left[R_1^l\text{Re}(u) - R_2^l\text{Im}(u)\right] + i\,\left[R_1^l\text{Im}(u) + R_2^l\text{Re}(u)\right], && (7)\\ u &\leftarrow \left[R_3^l\text{Re}(u) - R_4^l\text{Im}(u)\right] + i\,\left[R_3^l\text{Im}(u) + R_4^l\text{Re}(u)\right], && (8)\\ u &\leftarrow F^{-1}(u), && (9)\\ u &\leftarrow \sigma(u) + W_1^l\, u_{\text{orig}} + b_1^l, && (10) \end{aligned}$$

where F and F^{-1} are the Fourier and inverse Fourier transforms, and W_1^l are the weight tensors and b_1^l is the bias tensor for l^{th} layer. Re and Im represent the real and imaginary parts of a complex number. Note that expressions (6)—(9) are performed in series, that is, u is updated at every step in place. The role of these operations is to cast the tensor u from its original function space into a new function space modified by the learnable weights R_1^l, R_2^l, R_3^l, and R_4^l. Finally, in expression (10) we employ the Gaussian error linear unit (Hendrycks & Gimpel, 2016) non-linearity shown by σ and add back the original function multiplied by learnable weight W^l and bias b^l of shapes $C\times C$ and C, respectively. This residual connection allows the model to learn autocorrelations. Expressions (5)—(10) represent the spatial mixing process. The shape of u at the end of equation (10) is $C \times h \times w$.

Alternatively, if we assume after the Fourier transform in equation (6), $u = H\, e^{i\theta}$, equation (7) can be represented as

$$u \leftarrow R^l\, e^{i(\theta+\alpha)}, \tag{11}$$

where $R^l = H\,\sqrt{R_1^{l\,2} + R_2^{l\,2}}$, $\alpha = \tan^{-1}\left(R_2^l/R_1^l\right)$. This representation makes it clear that spatial mixing in equation (7) learns a mapping from a Fourier space (H, θ) to a new Fourier space $(R^l, \theta+\alpha)$. Similar mapping is learned in equation (8).

Traditional convolutional neural networks require deep networks to learn relationships between locations that are far apart. Performing the convolution in the wavenumber space alleviates this issue. Another advantage of the Fourier transform is that higher order Fourier modes can be ignored which further improves computational efficiency. For example, our horizontal dimensions $h \times w$ allow $\frac{h}{2} \times w$ Fourier modes but we only retain first $N_{\mathrm{h}} \times N_{\mathrm{w}}$ modes. Therefore, the weights R_1^l, R_2^l, R_3^l, and R_4^l have dimensions $N_{\mathrm{h}} \times N_{\mathrm{h}}$. In the mlp-mixer (Tolstikhin et al., 2021), spatial mixing is performed without the Fourier transform which does not allow truncation of higer-order Fourier modes. This limitation makes their spatial mixing step computationally expensive.

In the channel mixing phase, the C channels are transformed into a new set of C channels by multiplying by learnable weights and a non-linearity. Mathematically, this operation is represented by

$$\begin{aligned} u_{\mathrm{orig}} &\leftarrow u, && (12) \\ u &\leftarrow \sigma(W_2^l\, u + b_2^l), && (13) \\ u &\leftarrow W_3^l\, u + b_3^l, && (14) \\ u &\leftarrow u + W_4^l \odot u_{\mathrm{orig}} + b_4^l, && (15) \end{aligned}$$

where the weights W_2^l and W_3^l have dimensions $C_{\mathrm{hidden}} \times C$ and $C \times C_{\mathrm{hidden}}$, respectively. Biases, b_2^l and b_3^l, have dimensions C_{hidden} and C, respectively. The weight W_4^l and bias b_4^l with shapes C provide a residual pathway and facilitate autocorrelations like in the spatial mixing phase. Layer normalization is performed after spatial mixing as well as channel mixing.

Notice that there is a residual pathway that bypasses the spatial and channel mixing stages. The expectation is that the small scale variability lost due to discarded Fourier modes can be learned through residual connections. In the spirit of best practice in deep learning, these blocks are repeated where output from one block is sent as an input to the next block. For our run, we set the depth $L = 3$ where we find a good compromise between lowest loss and computational efficiency.

3.3.3 Projection

The tensor shown on the right in fig. 2 shows the state of the final tensor, u obtained by decoding the C channels into the required number of output channels C_{out} with a two layer fully-connected neural network called projection. Mathematically, the projection channel is represented as below:

$$\begin{aligned} u &\leftarrow \sigma(W_{\mathrm{proj},1}\, u + b_{\mathrm{proj},1}), && (16) \\ u &\leftarrow W_{\mathrm{proj},2}\, u + b_{\mathrm{proj},2}, && (17) \end{aligned}$$

where $W_{\mathrm{proj},1}$ and $W_{\mathrm{proj},2}$ have shapes $C \times C_{\mathrm{proj}}$ and $C_{\mathrm{out}} \times C_{\mathrm{proj}}$, respectively. Biases $b_{\mathrm{proj},1}$ and $b_{\mathrm{proj},2}$ have shapes C_{proj} and C_{out}, respectively.

3.3.4 Pretraining and Finetuning

The loss is defined as the mean squared error between the tensor of shape $C_{\mathrm{out}} \times h \times w$ and the ground truth tensor at a later instant Δt time later similar to Scher (2018). The neural operator can be represented concisely as

$$\hat{u}_{\mathrm{i+1}} = \mathcal{G}(u_{\mathrm{i}}), \tag{18}$$

where i is any instant and $i{+}1$ is an instant Δt time later and $\mathcal{G}$ represents the series of operations described in section 3.3. For our study Δt is set to 5, 10, and 30 days. Loss is defined as

$$L_{\mathrm{PT}} = L_2(u_{\mathrm{i+1}}, \hat{u}_{\mathrm{i+1}}) + 0.01\, L_1(u_{\mathrm{i+1}}, \hat{u}_{\mathrm{i+1}}), \tag{19}$$

where

$$L_2(u_{\rm j}, \hat{u}_{\rm j}) = \frac{1}{N} \sum_{b,h,w,c} (u_{\rm j} - \hat{u}_{\rm j})^2, \tag{20}$$

and

$$L_1(u_{\rm j}, \hat{u}_{\rm j}) = \frac{1}{N} \sum_{b,h,w,c} \mid u_{\rm j} - \hat{u}_{\rm j} \mid, \tag{21}$$

$N = b \times h \times w \times C$. Here h and w represent the number of gridpoints in latitude and longitude, respectively, and C is the number of co-dimensions ($C_{\rm out}$). The evaluation in eq. (18) is performed over b training samples. Note that $u_{\rm i+1}$ is the ground truth and $\hat{u}_{\rm i+1}$ is the FNO prediction from the initial condition $u_{\rm i}$. L_2 loss tends to amplify large errors while there is no such bias in the L_1 loss (Esmaeilzadeh et al., 2020). Therefore, it is a good practice to use a weighted combination of both. Minimizing the loss in eq. (19) over the training data gives an emulator with learned weights and biases that performs reasonably on validation data. This step is called the pretraining step.

Once the pretraining loss is minimized, we perform the finetuning step. In the finetuning step, the model is used to predict the next two timesteps from an initial time step, that is

$$\hat{u}_{\rm i+1} = \mathcal{G}(u_{\rm i}), \tag{22}$$

$$\hat{u}_{\rm i+2} = \mathcal{G}(\hat{u}_{\rm i+1}). \tag{23}$$

The loss is now defined as

$$L_{\rm FT} = L_2(u_{\rm i+1}, \hat{u}_{\rm i+1}) + L_2(u_{\rm i+2}, \hat{u}_{\rm i+2}) + L_1(u_{\rm i+1}, \hat{u}_{\rm i+1}) + L_1(u_{\rm i+2}, \hat{u}_{\rm i+2}). \tag{24}$$

Minimizing this loss gives a model that is finetuned to make multistep predictions. We expect that the finetuning stage allows the emulator to further reinforce conservation principles, thus rendering stability to long-term predictions. Pathak et al. (2022) employ this technique in their emulator of ERA5 atmospheric data, but their emulator blows up after about three weeks of predictions. Note that their stability horizons are shorter because their aim is to emulate the full-scale atmosphere without any idealizations. Perhaps the simplifying nature of idealization in our study makes our emulator stable. Yet another way to teach conservation principles to the emulator is via the method suggested by Weyn et al. (2019), that is to design the emulator to ingest two time steps and output two later timesteps. To summarize, the parameters used for FNO are given in table 2.

4 Description of circulation in the double gyre numerical simulations

In observations of the Atlantic Ocean there is a decreasing pattern of SSH from the low latitudes to high latitudes in the northern hemisphere (e.g. Fu et al. (2020)). A strong western boundary current flows along the east coast of the US and turns offshore at around the location where SSH changes sign. An idealized representation of the north Atlantic basin is given by the double gyre simulation shown in Fig. 3. It shows the mean streamfunction, SSH, and sea surface temperature (SST) for the double gyre control simulation (experiment 1 in table 1). The large-scale pattern in the middle panel is qualitatively similar to that in the first figure of Fu et al. (2020). Additionally, the streamfunction in panel shows that there are two gyres encapsulated by the regions of positive and negative streamfunctions. The gyre circulation around positive (negative) values is clockwise (counter-clockwise). On the western edge of these gyres, there are rapidly flowing western boundary currents. These currents are hotspots of mesoscale variablity as can be seen through the wiggles in streamfunction as well as SSH fields. The temperature pattern in panel c mimics the large-scale meridional SST gradient, but there is turbulent activity near the western boundary. The western boundary current and its offshoot can be thought of as analogous to the storm tracks in the atmosphere (Bryan, 1963). Scher and Messori (2019) noted that their emulators failed to correctly estimate the storm track locations in long climate runs. Our use of FNO seems to address this issue.

The above discussion of spatial patterns holds true for experiments 2 and 3 in table 1. Only difference from the control experiment is that the magnitudes of the gyre streamfunctions vary with wind forcing. Increasing wind speed leads to stronger gyre circulation and enhanced streamfunctions (not shown). We expect the emulator to learn this relationship between the ocean dynamics and wind forcing magnitude.

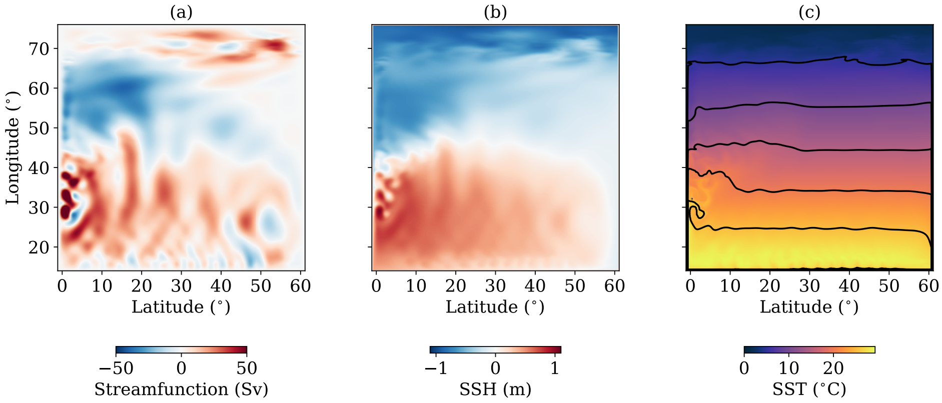


Figure 3. (a) Streamfunction ($10^6\,\mathrm{m}^3\,\mathrm{s}^{-1}$), (b) SSH anomaly (m), (c) SST (°C) in the control simulation at a randomly chosen instant.

5 Predictions with FNO

In this section, we compare the predictions made by the emulator with the ground truth for 15 randomly chosen initial conditions. Note that we use the simulation from MITgcm as the ground truth. The decorrelation time scale for this simulation is about three months. Therefore, we compare the FNO predictions against ground truth over two time scales — we compare the short-term prediction skill i.e. skill up to 100 days and long-term prediction skill i.e. skill up to 2000 days. Note that since 2000 days is well beyond the decorrelation time-scale of the simulation, we only expect statistics like the global mean of variables and kinetic energy spectrum to be conserved for a successful prediction. For the short-term prediction we expect the RMSE of the predicted time-series with the ground truth to be lower than that of the ground truth with climatology and persistence.

5.1 Short-term prediction skill

Similar to Rasp and Thuerey (2021), Scher and Messori (2019), and Pathak et al. (2022) we show the RMSE as a metric of the model skill. We choose surface speed, surface hydrostatic pressure anomaly (P/ρ_0), and surface temperature (T) as reasonable indicators of prediction skill. We focus mainly on the surface fields because, typically, in the oceans, largest variability is found near the surface, while the deep ocean tends to be quiescent. Nevertheless, the skills at deeper levels are comparable (not shown). In fig. 4, we compare the skill of FNO prediction with that of climatology and persistence. The climatology here is the pointwise time-mean of MITgcm simulation. We treat MITgcm simulation as the ground truth and calculate its RMSE with this mean. For calculating RMSE with respect to persistence, we assume the initial condition to be the prediction for all subsequent time steps. The RMSE between MITgcm simulation (ground

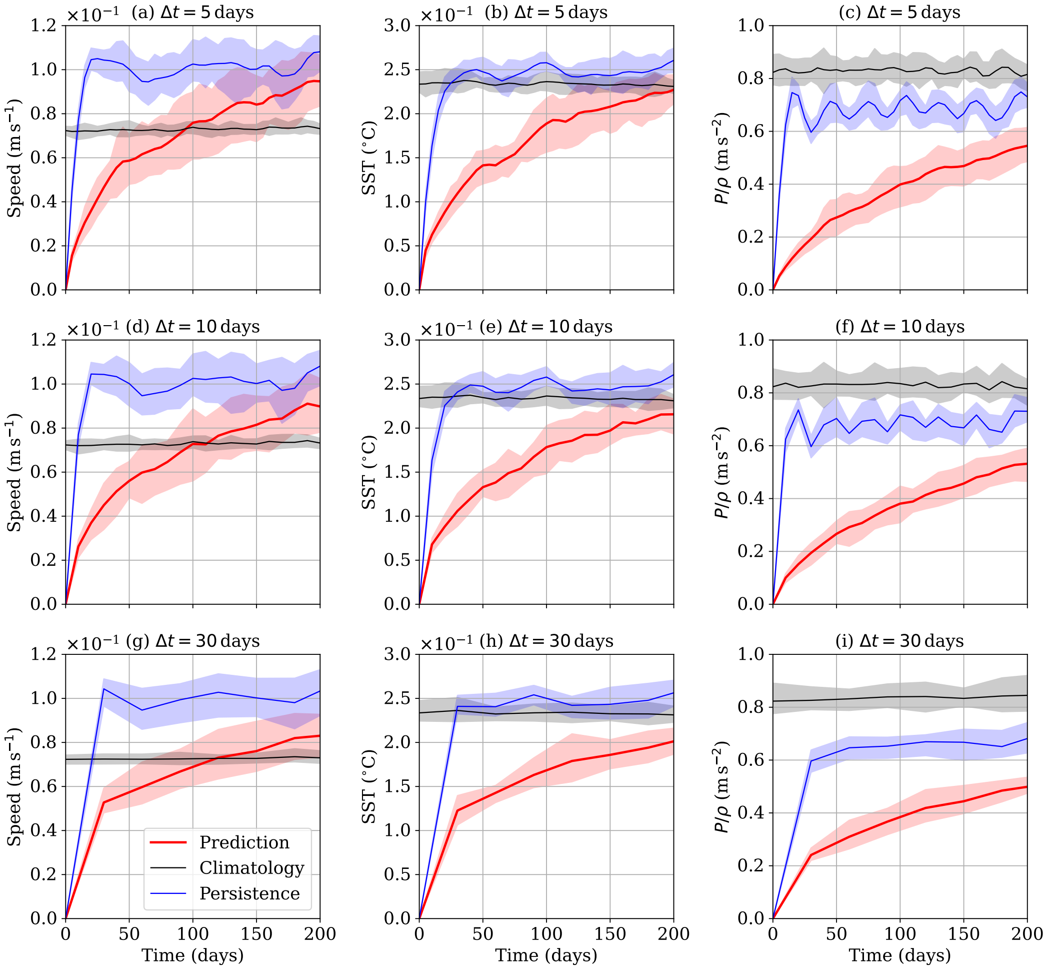


Figure 4. RMSE as a function of time for 15 ensembles with randomly chosen initial conditions from the testing dataset. Ensembles were integrated with $\Delta t = 5$ days, $\Delta t = 10$ days, and $\Delta t = 30$ days for the top, middle, and bottom rows, respectively. The first, second, and third columns represent surface speed ($\mathrm{m\,s^{-1}}$), surface pressure anomaly (P/ρ, $\mathrm{m\,s^{-2}}$), and SST (°C), respectively. The solid red lines represent the mean RMSE of predicted ensembles with ground truth, solid black lines represent the RMSE of ground truth ensembles with climatology, and solid blue lines represent the RMSE of ground truth ensembles with persistence. The shaded regions show the limits of the 10^{th} and 90^{th} percentiles obtained from 15 ensembles. Note that all the predicted ensembles here were obtained using the finetuned model.

truth) and the initial condition then gives the RMSE of persistence. In fig. 5 the RMSE of FNO prediction is lower than that of climatology and persistence thereby indicating that the FNO emulator is learning meaningful relationships from the training data. The RMSEs for velocities are lower than climatology and persistence up to the decorrelation time scale of about 3 months, while those for surface pressure and SST are lower for longer. This is expected because smale-scale features like eddies and sharp currents appear in velocity fields, while pressure and SST fields are dominated by large-scale equator-to-pole meridional gradients.

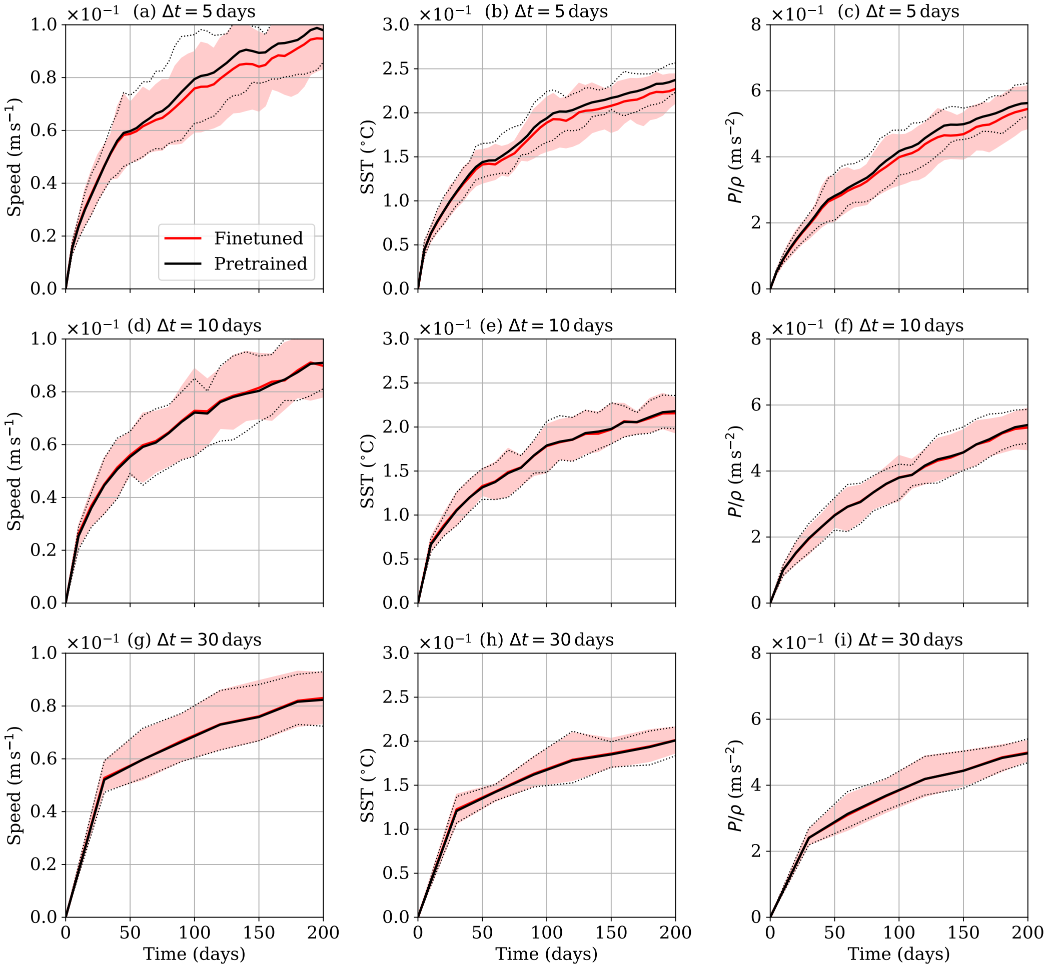


Figure 5. RMSE as a function of time for 15 ensembles with randomly chosen initial conditions from the testing dataset. Ensembles were integrated with $\Delta t = 5$ days, $\Delta t = 10$ days, and $\Delta t = 30$ days for the top, middle, and bottom rows, respectively. The first, second, and third columns represent surface speed ($\mathrm{m\,s^{-1}}$), surface pressure anomaly (P/ρ, $\mathrm{m\,s^{-2}}$), and SST (°C), respectively. The solid black lines represent the mean RMSE of ensembles integrated with the pretrained emulator, while the solid red lines represent the means of those integrated with finetuned emulator. The dotted black lines and the shaded red regions show the limits of the 10th and 90th percentiles obtained from 15 ensembles for pretrained and finetuned models, respectively.

Fig. 5 compares the RMSE of pretrained FNO and multistep-finetuned FNO with ground truth. We find that the RMSE of predictions made at 5-day intervals is more than

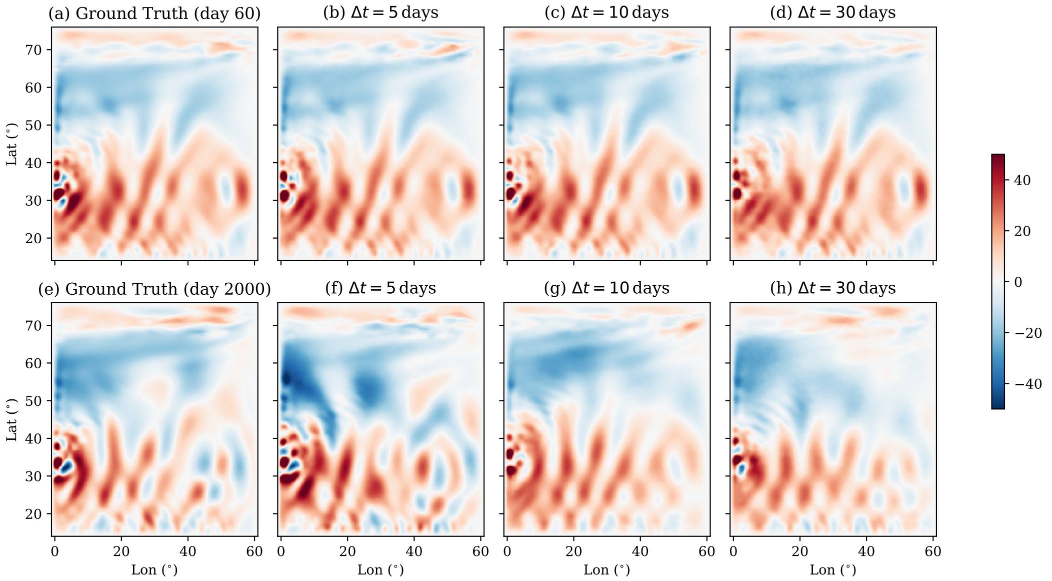


Figure 6. Streamfunction ($10^6\,\mathrm{m}^3\,\mathrm{s}^{-1}$) on the 60th day (top row) and 2000th day for the ground truth (first colum), and one FNO ensemble with prediction intervals of 5, 10 and 30 days (second, third, and fourth columns, respectively).

those made at 10-day and 30-day intervals. This finding is similar to that of Scher (2018) where their weather prediction emulator had the best accuracy at intermediate prediction intervals. Moreover, panels in the first row indicate that for the 5-day prediction interval, the RMSE of pretrained model is higher than that of the finetuned model. For the 10-day and 30-day prediction intervals (middle and bottom rows, respectively), the RMSE of pretrained and finetuned models is comparable. This indicates that multistep finetuning has a more profound effect on short than on intermediate and long prediction intervals. This shows that the lack of skill at short term prediction intervals arises from the basic fact that at short timescales the state of the underlying ocean variables does not change by much. Multistep finetuning ameliorates this limitation to some extent by effectively increasing the prediction interval (Li et al., 2021). Ideally, with more computational capability it would be worthwhile to perform multistep finetuning on more than two timesteps. On the longer end of prediction intervals (60 days and above, not shown), the state of the ocean might be too decorrelated for the model to learn meaningful predictability. Therefore, the model makes best predictions at intermediate intervals.

5.2 Long-term prediction skill

A desirable quality of numerical models, forced by a time-invariant forcing, is the maintenance of a statistically steady state, that is, global means of no predicted quantity should deviate by a large amount under steady forcing with forward integration. The numerical model usually achieves this through conserving quantities like mass and energy. Since, we don't explicitly prescribe these conservation laws to the emulator we hope that it learns them from the training data.

Fig. 6 compares the barotropic streamfunction predicted by iterative FNO ensembles from the same initial condition at 60th day and 2000th day. Two months is within the decorrelation scale of the experiment therefore we see that there is qualitative agreement between all panels in the top row. The panel c with 10-day prediction interval performs better especially near the western boundary where most of the turbulence is seen.

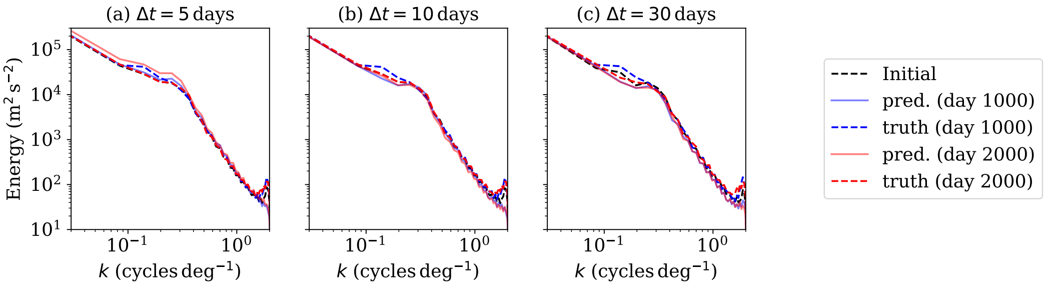


Figure 7. Kinetic energy spectrum as a function of wavenumber for the initial condition, 1000th day (blue), and 2000th day (red). Dotted and solid lines show the spectrums for MITgcm simulation (truth) and FNO predictions, respectively. First, second, and third panels use FNO models with prediction intervals of 5, 10, and 30 days, respectively.

On the 2000th day, however, all predictions are distinct from each other. This is expected because of the chaotic nature of the underlying phenomenon. Tiny differences in predictions within the decorrelation scale would grow as the iterative predictions timestep beyond the decorrelation scale. However, it is encouraging that the spatial scale of the anomalies is retained even after 2000 days of integration. This can be seen from fig. 7, where the FNO simulations retain the scales of variability even after 2000 days. We have also provided an animation of streamfunction as supplementary material from one ensemble with Δt of 10 days.

It should be noted that the 5-day prediction interval ensemble (fig. 6f) begins to get diffused around the 1800th day. This can also been seen in the energy spectrum (fig. 7a) where the energy in FNO prediction at small wavenumbers exceeds that in the ground truth. This limitation of the 5-day prediction interval stems from similar reasons as pointed out in the previous sections. The state of the ocean does not change much in 5 days. Therefore, we need extensive multistep finetuning. In the current study, we have only performed multistep training using two succesive timesteps. We perhaps need to use more timesteps to improve 5-day interval FNO emulator.

Fig. 8 shows the RMSE of surface speeds, surface pressure, and SST with respect to climatology and persistence. The projections with 5-day intervals produce significant errors, especially for surface pressure and SST after about 500 days of integration indicating that the conclusions from fig. 6 and fig. 7 are not limited to one ensemble.

6 Summary and discussion

In this study, we present an emulator that accurately predicts the forward evolution of the double gyre simulation. As far as we know, this is the first emulator that uses the state-of-the-art Fourier neural operators for ocean simulation. We specifically focus on the importance of prediction interval used to generate iterative forecasts. We show that the emulator stays stable for creating long ensembles at intermediate prediction intervals of about $O(10\,\text{days})$. Shorter prediction intervals require extensive multistep finetuning. Longer prediction intervals lie beyond the decorrelation scale of the experiment and fail to learn meaningful relationships in the underlying data.

A prominent feature of the emulator introduced here is its long-term stability. As far as we are aware, previous studies except Bonev et al. (2023) have failed to maintain the spectrum of variability in long prediction ensembles. These limitations stem from the nature of the neural networks used. CNNs, for example, are inherently diffusive. The

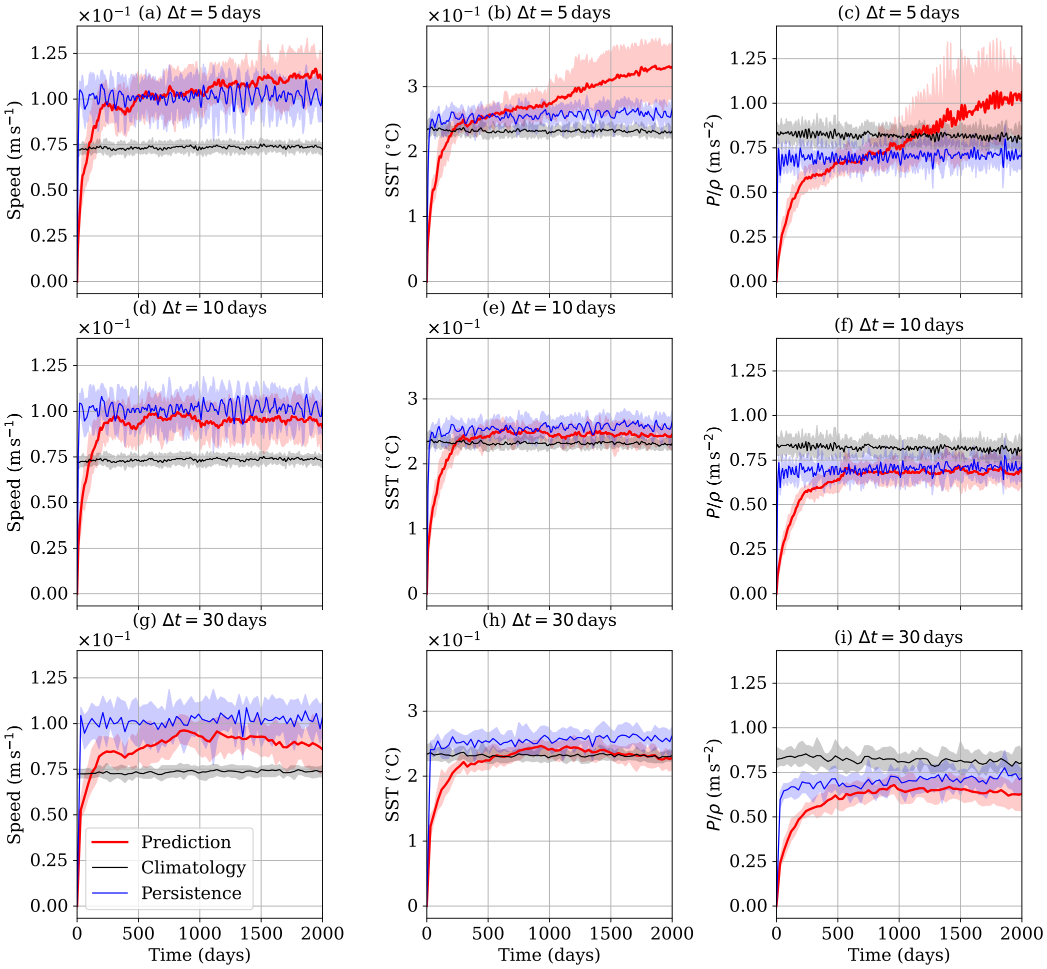


Figure 8. RMSE as a function of time for 15 ensembles with randomly chosen initial conditions from the testing dataset. Ensembles were integrated with $\Delta t = 5$ days, $\Delta t = 10$ days, and $\Delta t = 30$ days for the top, middle, and bottom rows, respectively. The first, second, and third columns represent surface speed ($\mathrm{m\,s^{-1}}$), surface pressure anomaly (P/ρ, $\mathrm{m\,s^{-2}}$), and SST (°C), respectively. The solid red lines represent the mean RMSE of predicted ensembles with ground truth, solid black lines represent the RMSE of ground truth ensembles with climatology, and solid blue lines represent the RMSE of ground truth ensembles with persistence. The shaded regions show the limits of the 10^{th} and 90^{th} percentiles obtained from 15 ensembles. Note that all the predicted ensembles here were obtained using the finetuned model.

most commonly used mean square error metric also tends to minimize errors only on large scales, thereby leading to smoother emulations. The Fourier neural operator used in this study overcomes these issues as it performs convolutions in the wavenumber space. This aspect has a profound effect on distant geographic locations that might be correlated. For traditional CNNs, the neural network has to be deep for two geographically distant locations to start exchanging information, while with the use of Fourier transform in FNO the information exchange takes place globally in the first layer itself of the neural network.

An additional feature of FNOs is that they are resolution-invariant. This allows us, for example, to train the network at higher resolution and make inferences at a lower resolution. We earmark exploiting this property of FNOs for a future study.

7 Open Research

The code for the emulator is available at `https://github.com/suyashbire1/oceanfourcast` (Bire & Lütjens, 2023).

The training and testing datasets used in this study will be uploaded to an open data-sharing platform before the review process is over.

Acknowledgments

MITgcm simulations used in this study were performed on the MIT Engaging cluster. Neural network training, testing, and inference was done on the MIT Satori cluster.

References

Bauer, P., Thorpe, A., & Brunet, G. (2015, Sep). The quiet revolution of numerical weather prediction. *Nature*, *525*(7567), 47–55. Retrieved from `http://dx.doi.org/10.1038/nature14956` doi: 10.1038/nature14956

Ben-Bouallegue, Z., Clare, M. C. A., Magnusson, L., Gascon, E., Maier-Gerber, M., Janousek, M., ... Pappenberger, F. (2023). *The rise of data-driven weather forecasting.* arXiv. Retrieved from `https://arxiv.org/abs/2307.10128` doi: 10.48550/ARXIV.2307.10128

Berloff, P., & Meacham, S. P. (1998, Feb). The dynamics of a simple baroclinic model of the wind-driven circulation. *Journal of Physical Oceanography*, *28*(2), 361–388. Retrieved from `http://dx.doi.org/10.1175/1520-0485(1998)028<0361:TDOASB>2.0.CO;2` doi: 10.1175/1520-0485(1998)028⟨0361:tdoasb⟩2.0.co;2

Bi, K., Xie, L., Zhang, H., Chen, X., Gu, X., & Tian, Q. (2022). *Pangu-Weather: A 3d high-resolution model for fast and accurate global weather forecast.* arXiv. Retrieved from `https://arxiv.org/abs/2211.02556` doi: 10.48550/ARXIV.2211.02556

Bire, S., & Lütjens, B. (2023, November). *suyashbire1/oceanfourcast: On zenodo.* Zenodo. Retrieved from `https://doi.org/10.5281/zenodo.10152441` doi: 10.5281/zenodo.10152441

Bolton, T., & Zanna, L. (2019, Jan). Applications of deep learning to ocean data inference and subgrid parameterization. *Journal of Advances in Modeling Earth Systems*, *11*(1), 376–399. Retrieved from `http://dx.doi.org/10.1029/2018MS001472` doi: 10.1029/2018ms001472

Bonev, B., Kurth, T., Hundt, C., Pathak, J., Baust, M., Kashinath, K., & Anandkumar, A. (2023). *Spherical fourier neural operators: Learning stable dynamics on the sphere.* arXiv. Retrieved from `https://arxiv.org/abs/2306.03838` doi: 10.48550/ARXIV.2306.03838

Bryan, K. (1963, Nov). A numerical investigation of a nonlinear model of a wind-driven ocean. *Journal of the Atmospheric Sciences*, *20*(6), 594–606. Retrieved from `http://dx.doi.org/10.1175/1520-0469(1963)020<0594:ANIOAN>2.0.CO;2` doi: 10.1175/1520-0469(1963)020⟨0594:anioan⟩2.0.co;2

Cachay, S. R., Erickson, E., Bucker, A. F. C., Pokropek, E., Potosnak, W., Bire, S., . . . Lütjens, B. (2021). *The world as a graph: Improving el niño forecasts with graph neural networks.* arXiv. Retrieved from `https://arxiv.org/abs/2104.05089` doi: 10.48550/ARXIV.2104.05089

Christensen, H., & Zanna, L. (2022, Dec). Parametrization in weather and climate models. *Oxford Research Encyclopedia of Climate Science*. Retrieved from `http://dx.doi.org/10.1093/acrefore/9780190228620.013.826` doi: 10.1093/acrefore/9780190228620.013.826

Cox, M. D., & Bryan, K. (1984, April). A numerical model of the ventilated thermocline. *Journal of Physical Oceanography*, *14*(4), 674–687. Retrieved from `http://dx.doi.org/10.1175/1520-0485(1984)014<0674:ANMOTV>2.0.CO;2` doi: 10.1175/1520-0485(1984)014⟨0674:anmotv⟩2.0.co;2

Dewitte, S., Cornelis, J. P., Müller, R., & Munteanu, A. (2021, Aug). Artificial intelligence revolutionises weather forecast, climate monitoring and decadal prediction. *Remote Sensing*, *13*(16), 3209. Retrieved from `http://dx.doi.org/10.3390/rs13163209` doi: 10.3390/rs13163209

Dueben, P. D., & Bauer, P. (2018, Oct). Challenges and design choices for global weather and climate models based on machine learning. *Geoscientific Model Development*, *11*(10), 3999–4009. Retrieved from `http://dx.doi.org/10.5194/gmd-11-3999-2018` doi: 10.5194/gmd-11-3999-2018

ECMWF charts. (n.d.). `https://charts.ecmwf.int/`. (Accessed: 2023-11-16)

Esmaeilzadeh, S., Azizzadenesheli, K., Kashinath, K., Mustafa, M., Tchelepi, H. A., Marcus, P., . . . others (2020). Meshfreeflownet: A physics-constrained deep continuous space-time super-resolution framework. In *Sc20: International conference for high performance computing, networking, storage and analysis* (pp. 1–15).

Fu, Y., Li, F., Karstensen, J., & Wang, C. (2020, Nov). A stable Atlantic meridional overturning circulation in a changing North Atlantic Ocean since the 1990s. *Science Advances*, *6*(48). Retrieved from `http://dx.doi.org/10.1126/sciadv.abc7836` doi: 10.1126/sciadv.abc7836

Guibas, J., Mardani, M., Li, Z., Tao, A., Anandkumar, A., & Catanzaro, B. (2021). *Adaptive Fourier neural operators: Efficient token mixers for transformers.* arXiv. Retrieved from `https://arxiv.org/abs/2111.13587` doi: 10.48550/ARXIV.2111.13587

Guillaumin, A. P., & Zanna, L. (2021, Sep). Stochastic-deep learning parameterization of ocean momentum forcing. *Journal of Advances in Modeling Earth Systems*, *13*(9). Retrieved from `http://dx.doi.org/10.1029/2021MS002534` doi: 10.1029/2021ms002534

Hendrycks, D., & Gimpel, K. (2016). *Gaussian error linear units (GELUs).* arXiv. Retrieved from `https://arxiv.org/abs/1606.08415` doi: 10.48550/ARXIV.1606.08415

Hersbach, H., Bell, B., Berrisford, P., Hirahara, S., Horányi, A., Muñoz-Sabater, J., . . . Thépaut, J. (2020, Jun). The ERA5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, *146*(730), 1999–2049. Retrieved from `http://dx.doi.org/10.1002/qj.3803` doi: 10.1002/qj.3803

Höhlein, K., Kern, M., Hewson, T., & Westermann, R. (2020, Nov). A comparative study of convolutional neural network models for wind field downscaling. *Meteorological Applications*, *27*(6). Retrieved from `http://dx.doi.org/10.1002/met.1961` doi: 10.1002/met.1961

Jiang, P., Yang, Z., Wang, J., Huang, C., Xue, P., Chakraborty, T. C., … Qian, Y. (2023, Jul). Efficient super-resolution of near-surface climate modeling using the fourier neural operator. *Journal of Advances in Modeling Earth Systems*, *15*(7). Retrieved from http://dx.doi.org/10.1029/2023MS003800 doi: 10.1029/2023ms003800

Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., & Anandkumar, A. (2021). *Neural operator: Learning maps between function spaces.* arXiv. Retrieved from https://arxiv.org/abs/2108.08481 doi: 10.48550/ARXIV.2108.08481

Kurth, T., Subramanian, S., Harrington, P., Pathak, J., Mardani, M., Hall, D., … Anandkumar, A. (2022). *FourCastNet: Accelerating global high-resolution weather forecasting using adaptive Fourier neural operators.* arXiv. Retrieved from https://arxiv.org/abs/2208.05419 doi: 10.48550/ARXIV.2208.05419

Lam, R., Sanchez-Gonzalez, A., Willson, M., Wirnsberger, P., Fortunato, M., Pritzel, A., … Battaglia, P. (2022). *GraphCast: Learning skillful medium-range global weather forecasting.* arXiv. Retrieved from https://arxiv.org/abs/2212.12794 doi: 10.48550/ARXIV.2212.12794

Lee-Thorp, J., Ainslie, J., Eckstein, I., & Ontanon, S. (2021). *FNet: Mixing tokens with Fourier transforms.* arXiv. Retrieved from https://arxiv.org/abs/2105.03824 doi: 10.48550/ARXIV.2105.03824

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., & Anandkumar, A. (2020a). *Fourier neural operator for parametric partial differential equations.* arXiv. Retrieved from https://arxiv.org/abs/2010.08895 doi: 10.48550/ARXIV.2010.08895

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., & Anandkumar, A. (2020b). *Neural operator: Graph kernel network for partial differential equations.* arXiv. Retrieved from https://arxiv.org/abs/2003.03485 doi: 10.48550/ARXIV.2003.03485

Li, Z., Liu-Schiaffini, M., Kovachki, N., Liu, B., Azizzadenesheli, K., Bhattacharya, K., … Anandkumar, A. (2021). *Learning dissipative dynamics in chaotic systems.* arXiv. Retrieved from https://arxiv.org/abs/2106.06898 doi: 10.48550/ARXIV.2106.06898

Marshall, J., Adcroft, A., Hill, C., Perelman, L., & Heisey, C. (1997, Mar). A finite-volume, incompressible Navier Stokes model for studies of the ocean on parallel computers. *Journal of Geophysical Research: Oceans*, *102*(C3), 5753–5766. Retrieved from http://dx.doi.org/10.1029/96JC02775 doi: 10.1029/96jc02775

Nguyen, T., Brandstetter, J., Kapoor, A., Gupta, J. K., & Grover, A. (2023). *ClimaX: A foundation model for weather and climate.* arXiv. Retrieved from https://arxiv.org/abs/2301.10343 doi: 10.48550/ARXIV.2301.10343

Pathak, J., Subramanian, S., Harrington, P., Raja, S., Chattopadhyay, A., Mardani, M., … Anandkumar, A. (2022). *FourCastNet: A global data-driven high-resolution weather model using adaptive Fourier neural operators.* arXiv. Retrieved from https://arxiv.org/abs/2202.11214 doi: 10.48550/ARXIV.2202.11214

Rao, Y., Zhao, W., Zhu, Z., Lu, J., & Zhou, J. (2021). *Global filter networks for image classification.* arXiv. Retrieved from https://arxiv.org/abs/2107.00645 doi: 10.48550/ARXIV.2107.00645

Rasp, S., Dueben, P. D., Scher, S., Weyn, J. A., Mouatadid, S., & Thuerey, N. (2020, Nov). Weatherbench: A benchmark data set for data-driven weather forecasting. *Journal of Advances in Modeling Earth Systems*, *12*(11). Retrieved from http://dx.doi.org/10.1029/2020MS002203 doi: 10.1029/2020ms002203

Rasp, S., & Thuerey, N. (2021, Feb). Data-driven medium-range weather prediction with a resnet pretrained on climate simulations: A new model for Weather-Bench. *Journal of Advances in Modeling Earth Systems*, *13*(2). Retrieved from `http://dx.doi.org/10.1029/2020MS002405` doi: 10.1029/2020ms002405

Scher, S. (2018, Nov). Toward data-driven weather and climate forecasting: Approximating a simple general circulation model with deep learning. *Geophysical Research Letters*, *45*(22), 12,616-12,622. Retrieved from `http://dx.doi.org/10.1029/2018GL080704` doi: 10.1029/2018gl080704

Scher, S., & Messori, G. (2019, Jul). Weather and climate forecasting with neural networks: Using general circulation models (GCMs) with different complexity as a study ground. *Geoscientific Model Development*, *12*(7), 2797–2809. Retrieved from `http://dx.doi.org/10.5194/gmd-12-2797-2019` doi: 10.5194/gmd-12-2797-2019

Tolstikhin, I., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., . . . Dosovitskiy, A. (2021). *MLP-Mixer: An all-MLP architecture for vision.* arXiv. Retrieved from `https://arxiv.org/abs/2105.01601` doi: 10.48550/ARXIV.2105.01601

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). *Attention is all you need.* arXiv. Retrieved from `https://arxiv.org/abs/1706.03762` doi: 10.48550/ARXIV.1706.03762

Wen, G., Li, Z., Azizzadenesheli, K., Anandkumar, A., & Benson, S. M. (2021). *U-FNO – an enhanced Fourier neural operator-based deep-learning model for multiphase flow.* arXiv. Retrieved from `https://arxiv.org/abs/2109.03697` doi: 10.48550/ARXIV.2109.03697

Weyn, J. A., Durran, D. R., & Caruana, R. (2019, Aug). Can machines learn to predict weather? Using deep learning to predict gridded 500-hpa geopotential height from historical weather data. *Journal of Advances in Modeling Earth Systems*, *11*(8), 2680–2693. Retrieved from `http://dx.doi.org/10.1029/2019MS001705` doi: 10.1029/2019ms001705

Weyn, J. A., Durran, D. R., & Caruana, R. (2020, Sep). Improving data-driven global weather prediction using deep convolutional neural networks on a cubed sphere. *Journal of Advances in Modeling Earth Systems*, *12*(9). Retrieved from `http://dx.doi.org/10.1029/2020MS002109` doi: 10.1029/2020ms002109

Yuval, J., & O'Gorman, P. A. (2023, Apr). Neural-network parameterization of subgrid momentum transport in the atmosphere. *Journal of Advances in Modeling Earth Systems*, *15*(4). Retrieved from `http://dx.doi.org/10.1029/2023MS003606` doi: 10.1029/2023ms003606

Zanna, L., & Bolton, T. (2020, Aug). Data-driven equation discovery of ocean mesoscale closures. *Geophysical Research Letters*, *47*(17). Retrieved from `http://dx.doi.org/10.1029/2020GL088376` doi: 10.1029/2020gl088376

Zhou, L., & Zhang, R.-H. (2023, Mar). A self-attention–based neural network for three-dimensional multivariate modeling and its skillful ENSO predictions. *Science Advances*, *9*(10). Retrieved from `http://dx.doi.org/10.1126/sciadv.adf2827` doi: 10.1126/sciadv.adf2827

Figure 1.

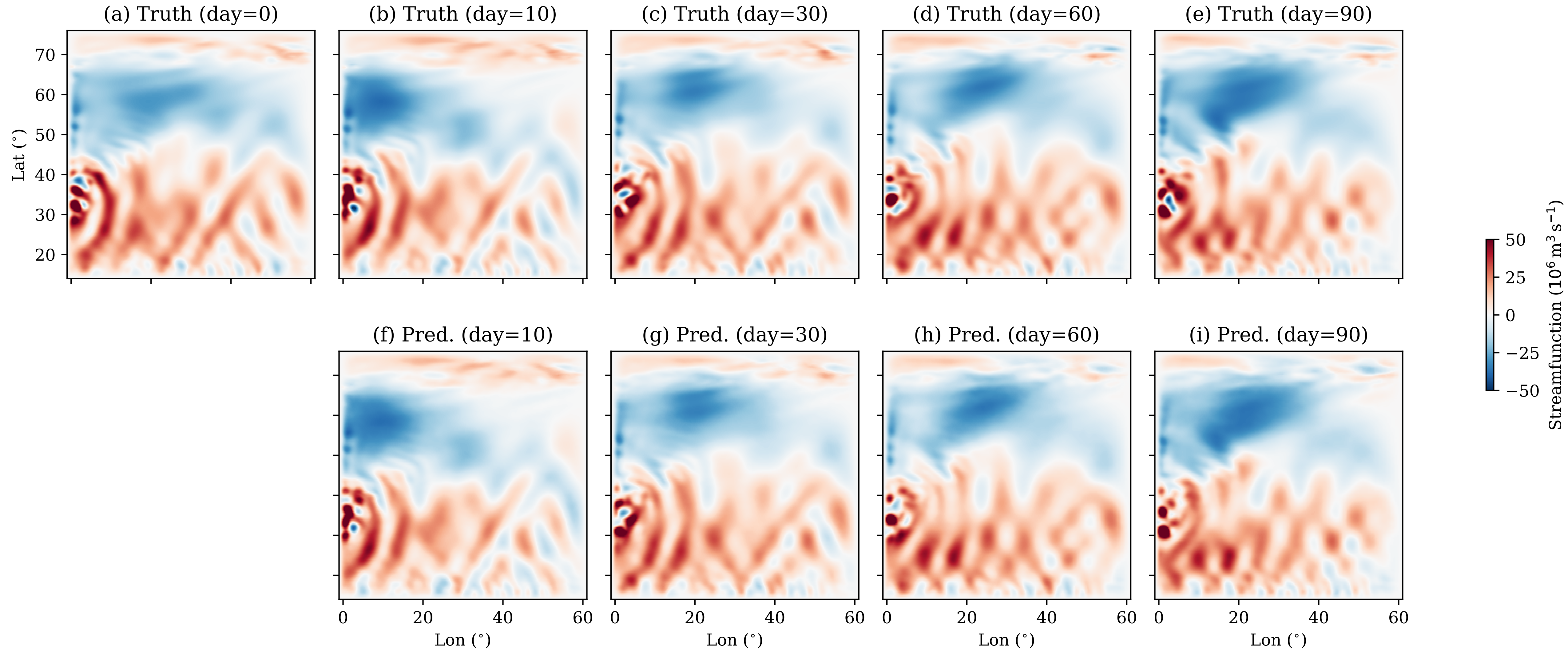

(a) Truth (day=0)
(b) Truth (day=10)
(c) Truth (day=30)
(d) Truth (day=60)
(e) Truth (day=90)
(f) Pred. (day=10)
(g) Pred. (day=30)
(h) Pred. (day=60)
(i) Pred. (day=90)
Lat (°)
70
60
50
40
30
20
0
20
40
60
Lon (°)
Streamfunction ($10^6\,\mathrm{m^3\,s^{-1}}$)
50
25
0
−25
−50

Figure 2.

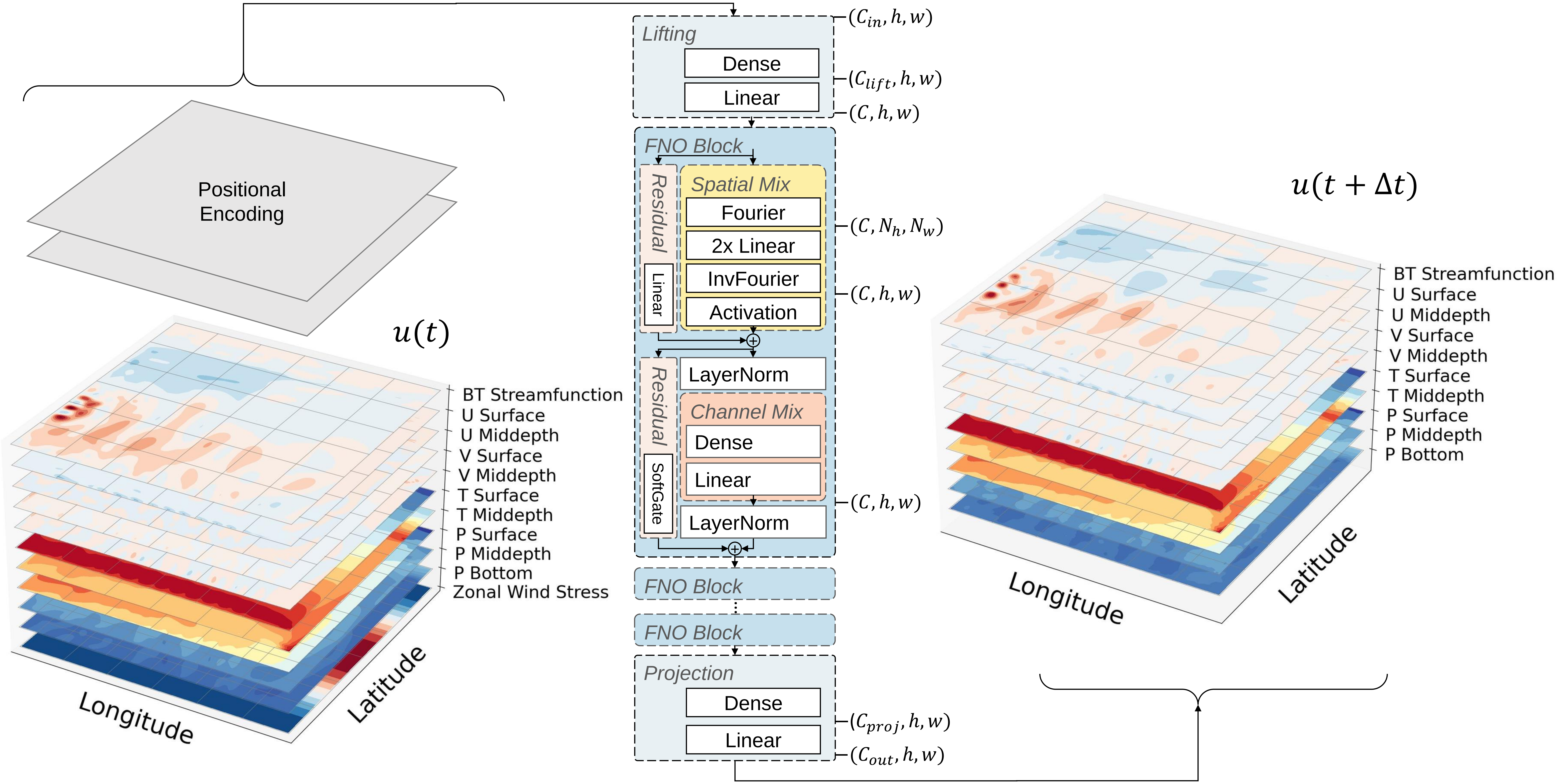
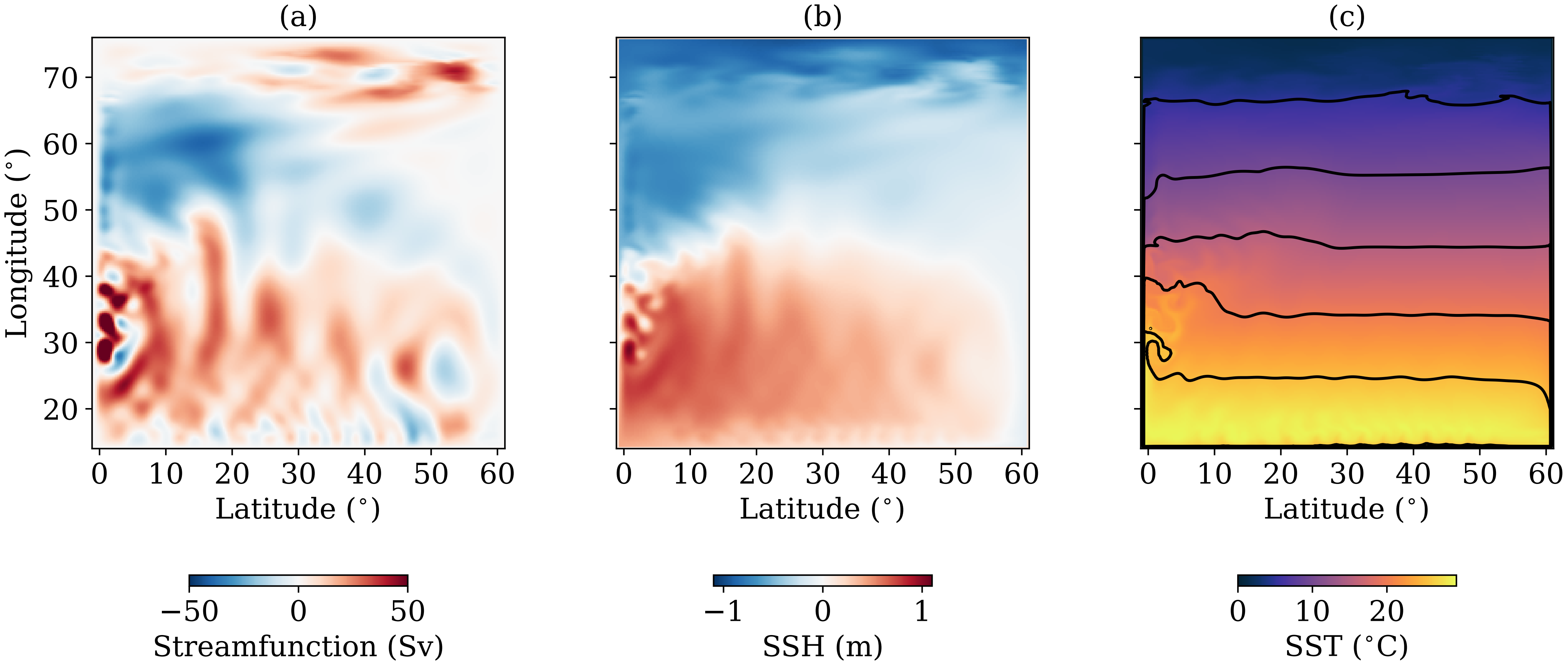
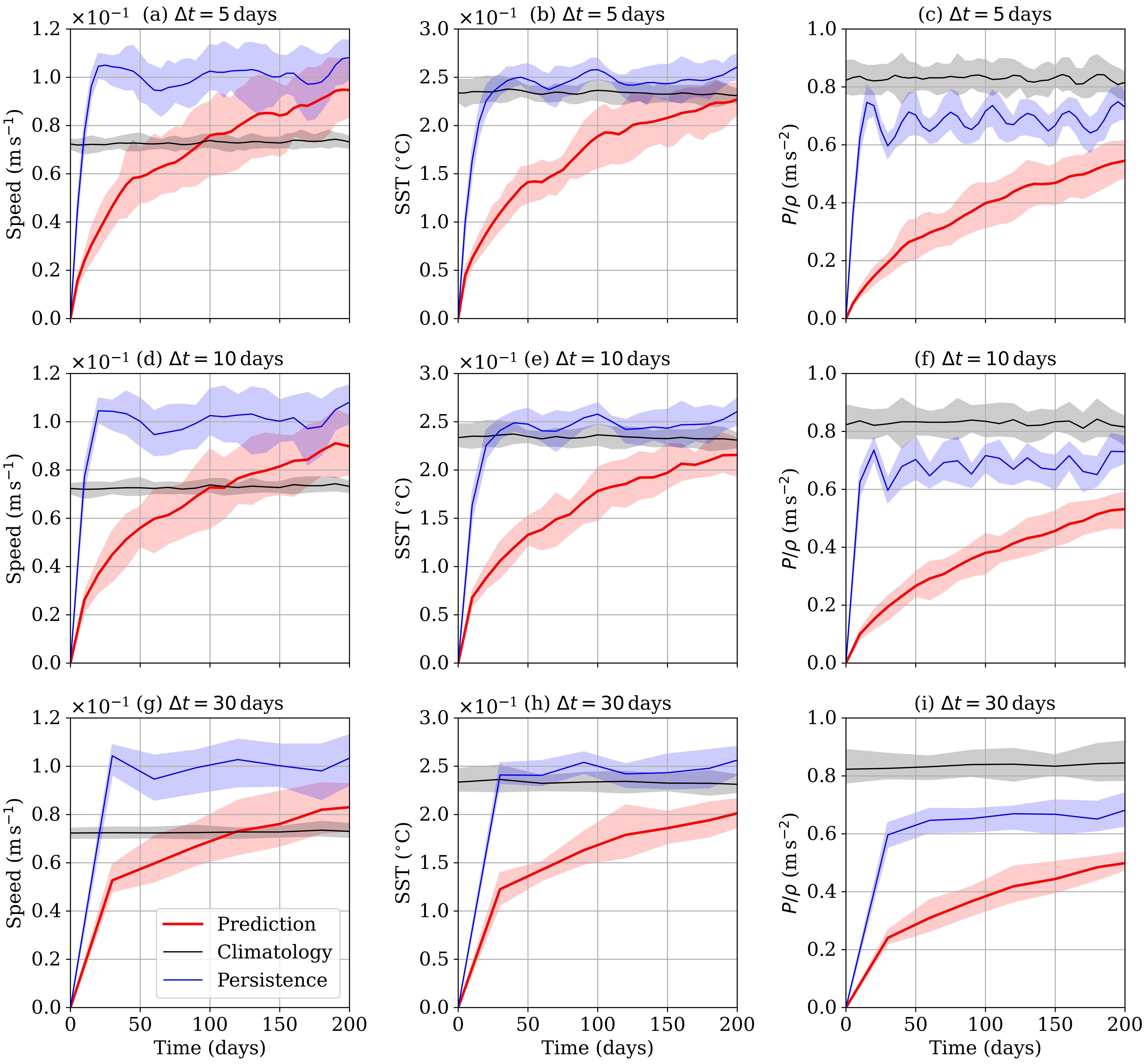

Positional
Encoding
$u(t)$
BT Streamfunction
U Surface
U Middepth
V Surface
V Middepth
T Surface
T Middepth
P Surface
P Middepth
P Bottom
Zonal Wind Stress
Longitude
Latitude
(C_{in}, h, w)
Lifting
Dense
(C_{lift}, h, w)
Linear
(C, h, w)
FNO Block
Residual
Linear
Spatial Mix
Fourier
(C, N_h, N_w)
2x Linear
InvFourier
(C, h, w)
Activation
Residual
SoftGate
LayerNorm
Channel Mix
Dense
Linear
(C, h, w)
LayerNorm
FNO Block
FNO Block
Projection
Dense
(C_{proj}, h, w)
Linear
(C_{out}, h, w)
$u(t + \Delta t)$
BT Streamfunction
U Surface
U Middepth
V Surface
V Middepth
T Surface
T Middepth
P Surface
P Middepth
P Bottom
Longitude
Latitude

Figure 3.



(a)
(b)
(c)
Longitude (°)
70
60
50
40
30
20
0
10
20
30
40
50
60
Latitude (°)
Latitude (°)
Latitude (°)
−50
0
50
Streamfunction (Sv)
−1
0
1
SSH (m)
0
10
20
SST (°C)

Figure 4.



×10⁻¹ (a) Δt = 5 days
×10⁻¹ (b) Δt = 5 days
(c) Δt = 5 days
×10⁻¹ (d) Δt = 10 days
×10⁻¹ (e) Δt = 10 days
(f) Δt = 10 days
×10⁻¹ (g) Δt = 30 days
×10⁻¹ (h) Δt = 30 days
(i) Δt = 30 days
Speed (m s⁻¹)
SST (°C)
P/ρ (m s⁻²)
Time (days)
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0.5
1.5
2.0
2.5
3.0
0
50
100
150
200
Prediction
Climatology
Persistence

Figure 5.

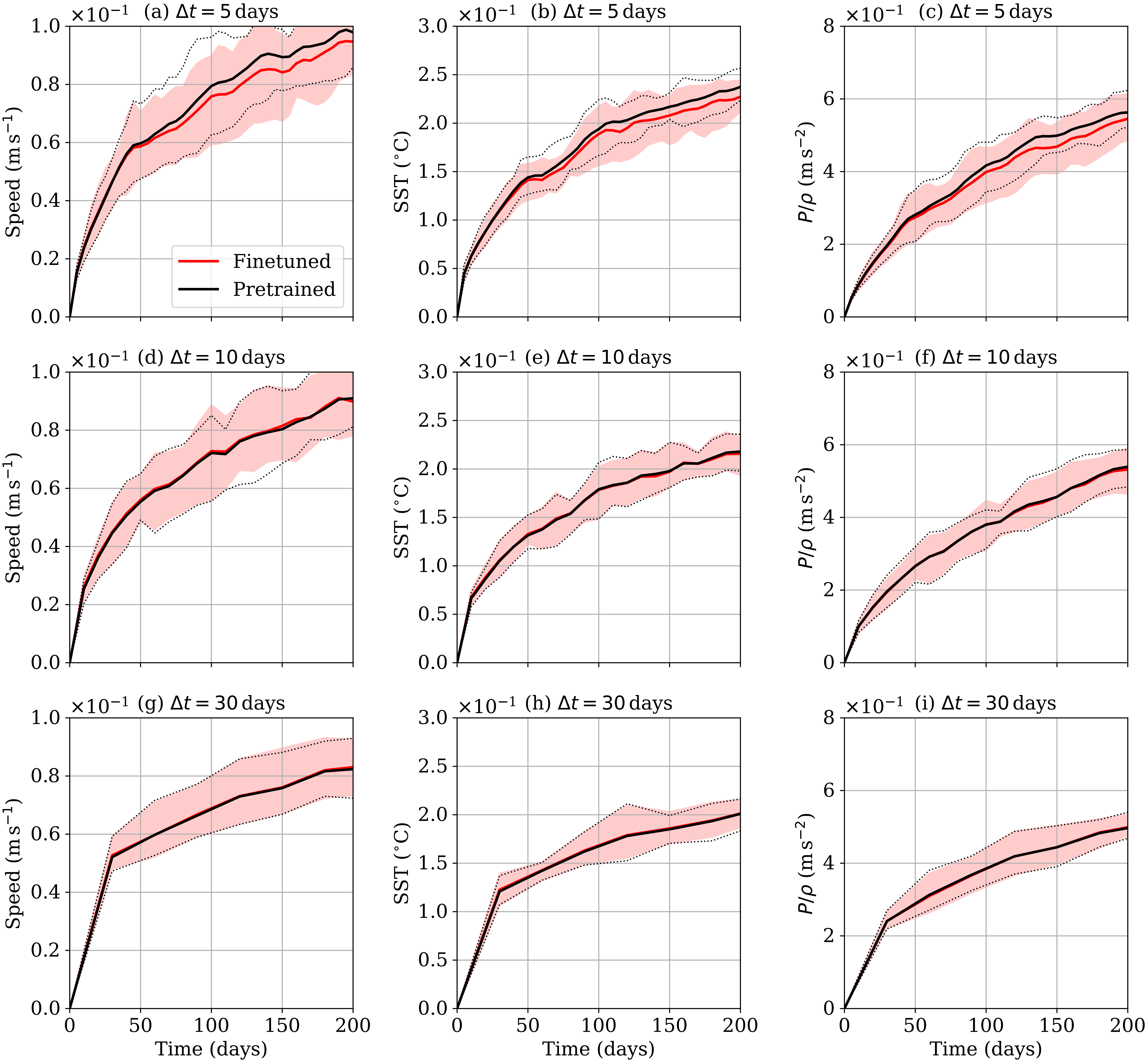

×10⁻¹ (a) Δt = 5 days
×10⁻¹ (b) Δt = 5 days
×10⁻¹ (c) Δt = 5 days
×10⁻¹ (d) Δt = 10 days
×10⁻¹ (e) Δt = 10 days
×10⁻¹ (f) Δt = 10 days
×10⁻¹ (g) Δt = 30 days
×10⁻¹ (h) Δt = 30 days
×10⁻¹ (i) Δt = 30 days
Speed (m s⁻¹)
SST (°C)
P/ρ (m s⁻²)
Time (days)
Finetuned
Pretrained
0.0
0.2
0.4
0.6
0.8
1.0
0.5
1.0
1.5
2.0
2.5
3.0
0
2
4
6
8
50
100
150
200

Figure 6.

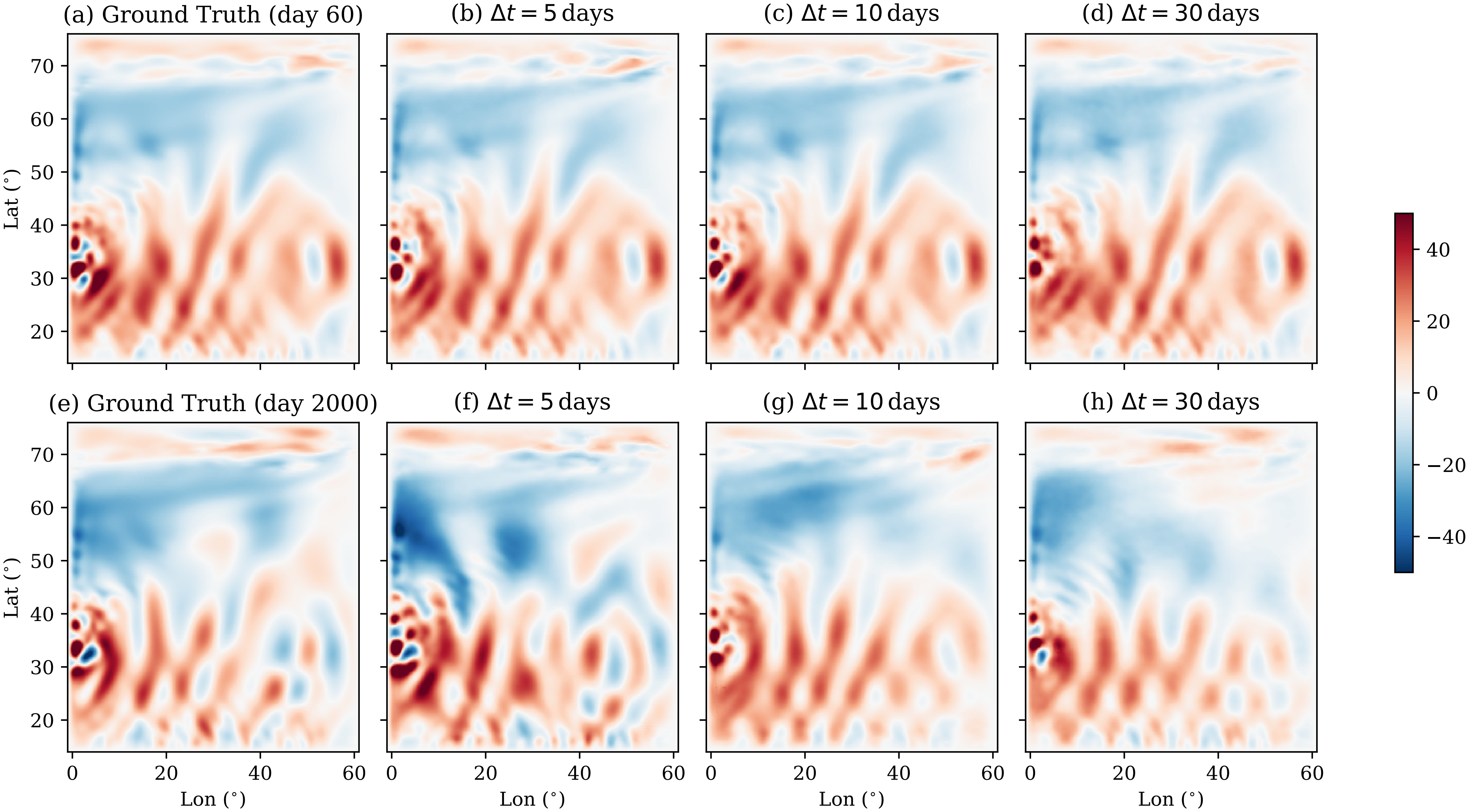

(a) Ground Truth (day 60)
(b) $\Delta t = 5\,\text{days}$
(c) $\Delta t = 10\,\text{days}$
(d) $\Delta t = 30\,\text{days}$
(e) Ground Truth (day 2000)
(f) $\Delta t = 5\,\text{days}$
(g) $\Delta t = 10\,\text{days}$
(h) $\Delta t = 30\,\text{days}$
Lat (°)
70
60
50
40
30
20
Lon (°)
0
20
40
60
40
20
0
−20
−40

Figure 7.

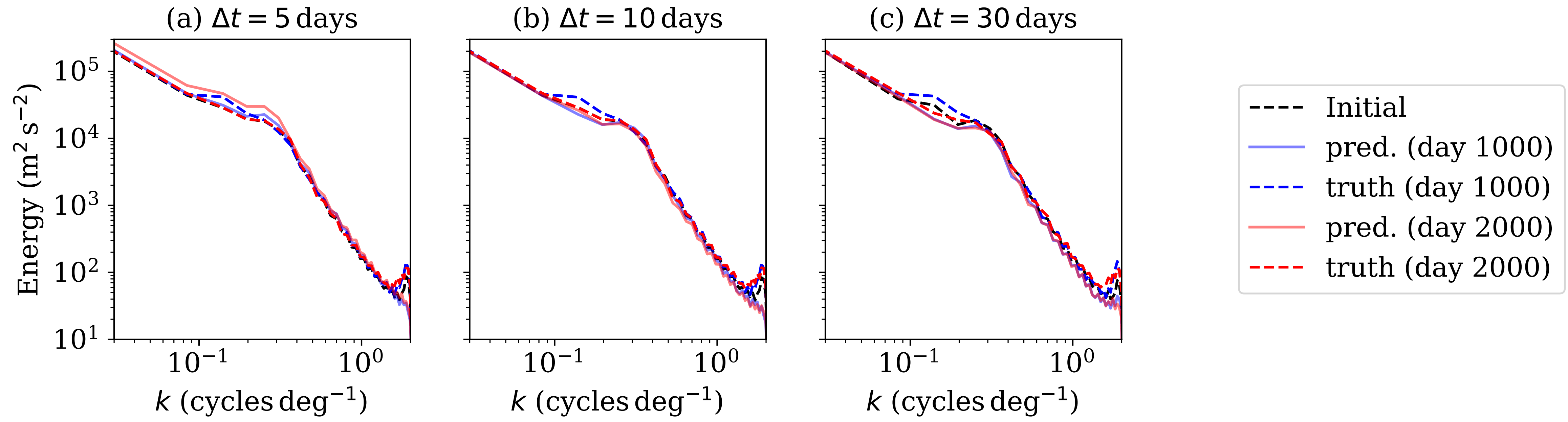
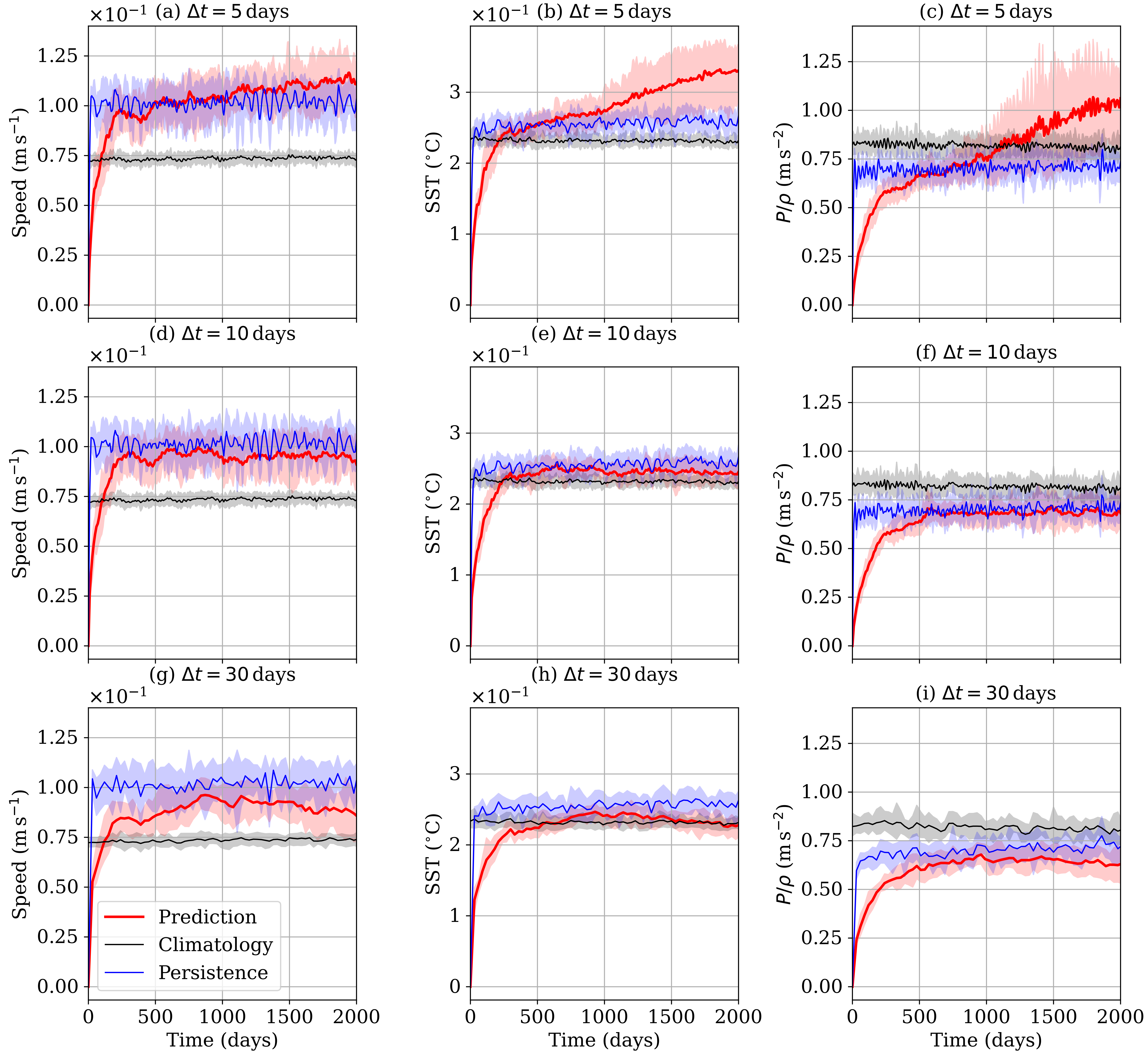
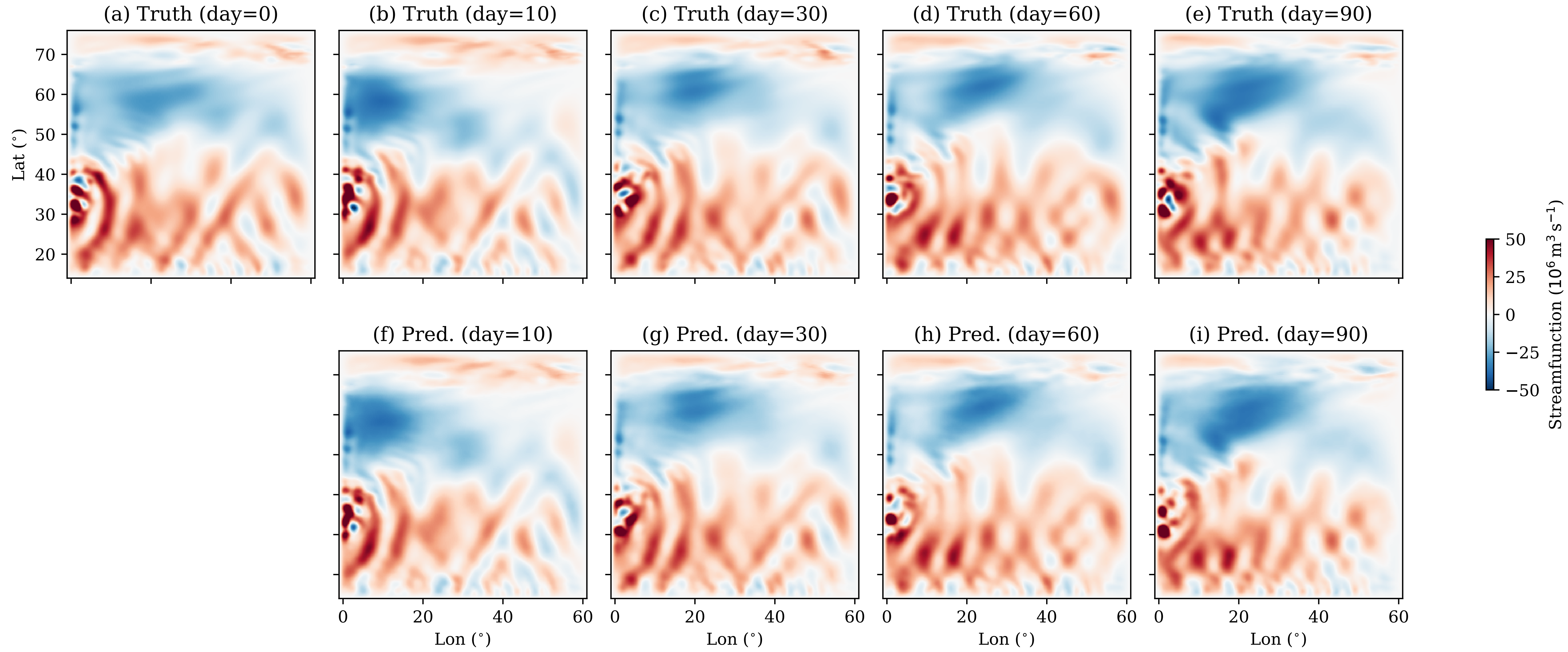
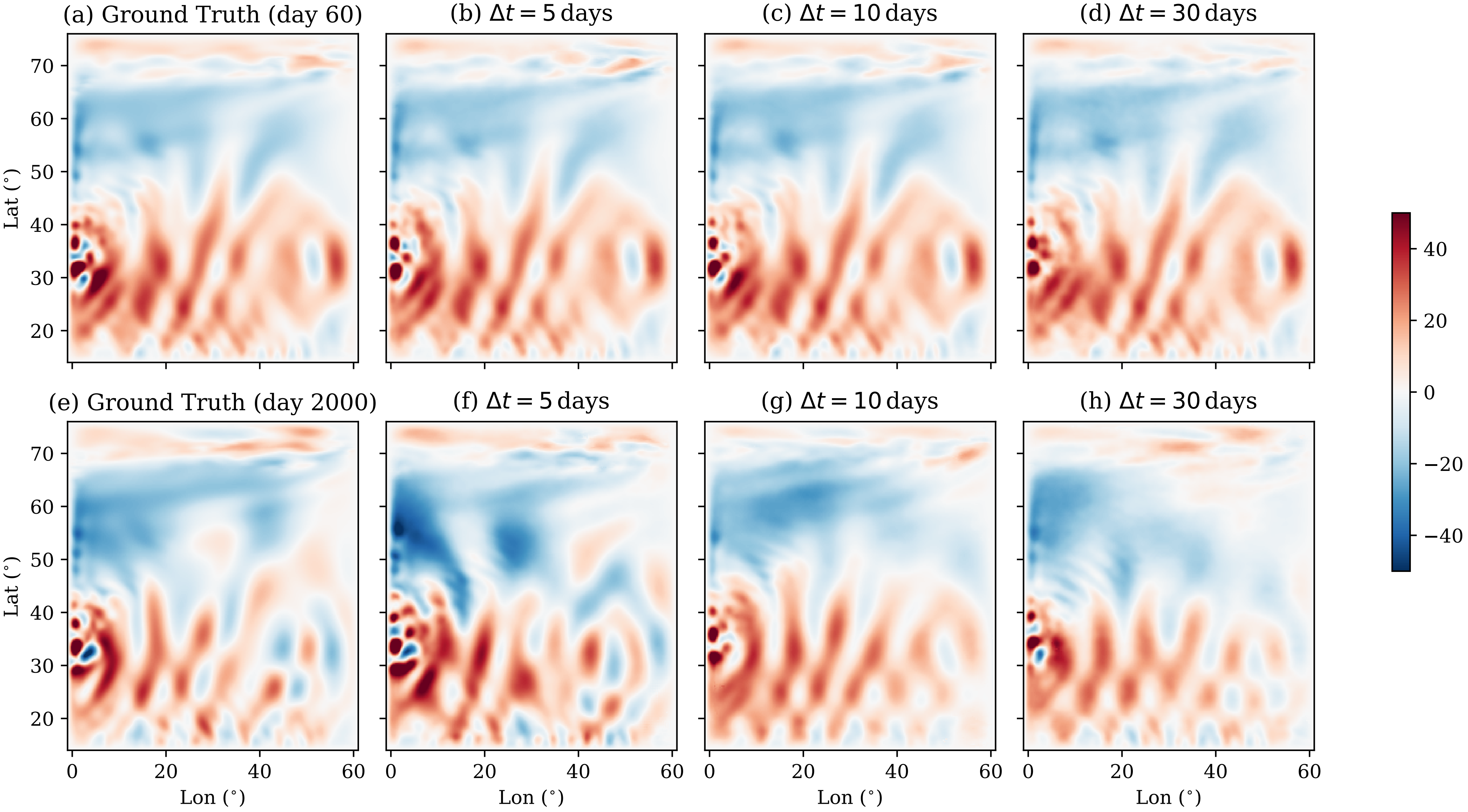

(a) $\Delta t = 5$ days
(b) $\Delta t = 10$ days
(c) $\Delta t = 30$ days
Energy ($\mathrm{m^2\,s^{-2}}$)
10^1
10^2
10^3
10^4
10^5
10^{-1}
10^0
k (cycles deg^{-1})
Initial
pred. (day 1000)
truth (day 1000)
pred. (day 2000)
truth (day 2000)

Figure 8.


(a) $\Delta t = 5$ days
(b) $\Delta t = 5$ days
(c) $\Delta t = 5$ days
(d) $\Delta t = 10$ days
(e) $\Delta t = 10$ days
(f) $\Delta t = 10$ days
(g) $\Delta t = 30$ days
(h) $\Delta t = 30$ days
(i) $\Delta t = 30$ days
$\times 10^{-1}$
Speed ($\mathrm{m\,s^{-1}}$)
SST (°C)
P/ρ ($\mathrm{m\,s^{-2}}$)
0.00
0.25
0.50
0.75
1.00
1.25
0
1
2
3
0
500
1000
1500
2000
Time (days)
Prediction
Climatology
Persistence


(a) Truth (day=0)
(b) Truth (day=10)
(c) Truth (day=30)
(d) Truth (day=60)
(e) Truth (day=90)
(f) Pred. (day=10)
(g) Pred. (day=30)
(h) Pred. (day=60)
(i) Pred. (day=90)
Lat (°)
70
60
50
40
30
20
0
20
40
60
Lon (°)
50
25
0
−25
−50
Streamfunction ($10^6\,\mathrm{m^3\,s^{-1}}$)


(a) Ground Truth (day 60)
(b) $\Delta t = 5\,\text{days}$
(c) $\Delta t = 10\,\text{days}$
(d) $\Delta t = 30\,\text{days}$
(e) Ground Truth (day 2000)
(f) $\Delta t = 5\,\text{days}$
(g) $\Delta t = 10\,\text{days}$
(h) $\Delta t = 30\,\text{days}$
Lat (°)
20
30
40
50
60
70
Lon (°)
0
20
40
60
40
20
0
−20
−40