




RESEARCH ARTICLE

10.1029/2025JH000655

Neural Networks Predicting Submesoscale Tracer Dispersion

Mayank Kumar Bijay^{1,2} and Jim Thomas^{1,2} ¹International Centre for Theoretical Sciences, Tata Institute of Fundamental Research, Bangalore, India, ²Centre for Applicable Mathematics, Tata Institute of Fundamental Research, Bangalore, India**Key Points:**

- We use neural network models to predict passive tracer fields directly from flow fields across a broad range of Rossby numbers
- Popular convolution based neural network architectures are seen to miss small-scale features in the predictions
- A new neural network architecture is developed and it successfully predicts submesoscale tracer dispersion features very well

Supporting Information:

Supporting Information may be found in the online version of this article.

Correspondence to:J. Thomas,
jimthomas.edu@gmail.com**Citation:**Bijay, M. K., & Thomas, J. (2025). Neural networks predicting submesoscale tracer dispersion. *Journal of Geophysical Research: Machine Learning and Computation*, 2, e2025JH000655. <https://doi.org/10.1029/2025JH000655>

Received 21 FEB 2025

Accepted 10 JUN 2025

Abstract In this paper we examine the possibility of directly predicting passive tracer dynamics from flow fields. We use a two-vertical-mode model for generating flows ranging from low to high Rossby numbers and advect the passive tracer with such flows. While typically tracer dynamics are obtained by time integrating a tracer advection equation, here the emphasis is on predicting the tracer dynamics from the flow field. We experiment with popular architectures such as Autoencoder, UNet, and GAN, and also develop a novel model, that we call LoConv, to make predictions. The LoConv model uses custom *Local Convolution* layers that allows convolution with spatially varying weights and this model outperforms usually used architectures such as Autoencoder, UNet, and GAN based on various metrics. Autoencoders with very few trainable parameters were unsuccessful in making good predictions even at large-scales. GAN and UNet predictions were biased towards large-scale features, unfavorable for capturing small-scale tracer dispersion, especially at high Rossby numbers. Overall, the LoConv model with some physics-informed training produced the best fine-scale tracer predictions, along with tracer flux and related derived quantities. More broadly, the results of this study point towards successful direct ways of predicting tracer fields from the flow, overcoming the computational cost of long numerical integration of tracer advection equations.

Plain Language Summary Oceanic tracers such as dissolved oxygen and carbon are important from a climate and ecological perspective. Typically these tracers are obtained computationally by integrating differential equations along with the flow in ocean models, which is computationally expensive and time-consuming. In this study we explore the possibility of using deep learning tools to make predictions about tracers. We trained neural network models to predict tracers from flow generated in different regimes, similar to oceanic flows whose scales vary from hundreds of kilometers to a kilometer. We experiment with some popular architectures and also develop some new neural net algorithms to make these predictions, with the aim of accurately predicting fine-scale tracer features. This makes analysis of tracer fields fast and highly efficient, by instantaneously obtaining tracer predictions instead of waiting for long computations. With these experiments, we come to a better understanding of the strengths and weaknesses of the various neural network architectures. Our developed model is particularly seen to make excellent fine-scale tracer predictions, outperforming popular architectures in various metrics.

1. Introduction

The transport and mixing of oceanic tracers such as heat, salinity, or dissolved nutrients like oxygen, carbon, etc. is determined by the oceanic flow. Oceanic tracers play a key role in ocean ecology, climate change, and the economy. More than 10% of ocean volume has hypoxic water (Bianchi et al., 2012) or water with ecologically stressful low concentration of oxygen, and so the transport and ventilation of oxygen tracer become important for the survival of species with such habitats. Ocean ventilation and transport of oxygen were studied by Gnanadesikan et al. (2013), where they looked at the effect of changing tracer diffusivity on the vertical and lateral mixing of oxygen. Another anthropogenic tracer, carbon, plays an important role for both marine life and climate. Oceans absorb a quarter of the total carbon produced by humans, slowing down global warming at the cost of decreasing pH and disturbing chemical balance in the ocean (Friedlingstein et al., 2022). The relationship between ocean flow and uptake of carbon was studied by Gnanadesikan et al. (2015). They highlight the importance of spatial variation of tracer diffusivity and the impact of increasing carbon dioxide levels on ocean carbon uptake. Similar studies have explored the uptake of other tracers that impact climate, such as chlorofluorocarbons for example, (Booth & Kamenkovich, 2008). More recently, Chouksey et al. (2022) showed that an increase in the isopycnal diffusivity, that is, diffusivity along fixed density surfaces, weakens both Atlantic residual overturning

© 2025 The Author(s). *Journal of Geophysical Research: Machine Learning and Computation* published by Wiley Periodicals LLC on behalf of American Geophysical Union.

This is an open access article under the terms of the [Creative Commons Attribution License](https://creativecommons.org/licenses/by/4.0/), which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

circulation as well as the Antarctic Circumpolar Current. Further connections between isopycnal diffusivity and different climate indices such as El Niño Southern Oscillations, North Pacific Gyre Oscillation, Atlantic Multidecadal Oscillation, Pacific Decadal Oscillation, and Dipole Mode Index were pointed out by Busecke and Abernathy (2019). These broad sets of studies emphasize the need to better understand and model the dispersion of various kinds of oceanic tracers, with an eye on improving their parametrizations in coarse resolution models.

The stirring action of oceanic flows on tracers is closely dependent on the scale at consideration. Horizontal length scales ranging from $O(100)$ k.m. to $O(10)$ k.m. are referred to as mesoscales. The geostrophically and hydrostatically balanced eddies in this scale regime contain about 90% of the oceanic flow kinetic energy (Ferrari & Wunsch, 2009; Richardson, 2008; Warren & Wunsch, 1981). Consequently, lateral stirring of tracers at mesoscales have been extensively studied using in situ tracer release and float dispersion experiments, satellite altimetry, and ocean models (Abernathy & Marshall, 2013; Holloway, 1986; J. R. Ledwell et al., 1993; Klocker & Abernathy, 2014; LaCasce et al., 2014; J. Ledwell et al., 2000; Zhurbas & Oh, 2004). Scales smaller than 10 k.m. falls within the regime of submesoscale and are characterized by departure in their dynamics from balance constraints. Internal gravity waves and other unbalanced flow components can energetically interact and generate fine-scale flow structures that cascades to smaller dissipative scales (McWilliams, 2016; Taylor & Thompson, 2023; Thomas, 2023). In situ observations, idealized and ocean model based studies reveal an increase in Rossby number and strength of the forward cascade of flow energy as we move from mesoscales to smaller submesoscales (Capet et al., 2008; Thomas & Vishnu, 2022; Thompson et al., 2016; Yu et al., 2019). Additionally, passive tracers, such as spice for example, are seen to have a steeper spectra as we transition from meso to submesoscales, this being an indication of more efficient stirring and dispersion of tracers due to the active role of unbalanced flow components (Klymak et al., 2015; Shcherbina et al., 2015; Spiro Jaeger et al., 2020; Sundermeyer et al., 2020).

As is clear from above discussion, dynamics of dispersion changes drastically as we go from mesoscales to submesoscales, making it challenging to capture tracer dispersion dynamics using coarse resolution models. Capturing fine scale submesoscale features in itself is difficult in coarse resolution models. Additionally, capturing tracer dispersion requires numerically integrating tracer equations advected by the flow. The computational cost increases when we consider realistic situations where dozens of tracers need to be advected at the same time. For example, Heuzé et al. (2023) lists some of the passive trace gases in the Arctic collected during the Multidisciplinary drifting Observatory for the Study of Arctic Climate (MOSAiC) expedition. In this expedition alone, they sampled two isotopes of helium, hydrogen, sulfur hexafluoride, tritium, and chlorofluorocarbon and considered their unsampled equivalents, such as carbon and argon. Raimondi et al. (2024) used radionuclides ^{129}I and ^{236}U to investigate Atlantic ventilation into the Arctic. These, along with some of the above-mentioned tracers, take the total number of tracers to over a dozen. Naturally we may ask, would it be possible to obtain the tracer dynamics directly from the flow without integrating separate partial differential evolution equations? In this paper we explore this possibility in an idealized set up using neural networks.

The development of machine learning algorithms and an expansion in affordable computational resources in recent years has led to various data-driven methods being applied to ocean and climate modeling. In particular, neural network models based on the framework of convolutional layers have gained popularity due to their ability to capture spatial features (Goodfellow, 2016; LeCun et al., 2015). A common use of convolution based models in oceanography is for subgrid parametrization or resolution upscaling. Bolton and Zanna (2019), for example, used a simple convolutional neural network to successfully model the subgrid scale momentum forcing. They also used the same architecture to predict subsurface streamfunctions from the surface layer streamfunction in a three-layer QG ocean model. More complicated architectures of autoencoders and generative adversarial networks (GANs) have been used for stochastic subgrid parametrization (Perezhogin et al., 2023). In these examples we see one physical field passing through a neural network to generate another physical field. Tracer prediction from flow field would also fall under this category of use of neural networks.

Neural networks have also been used for segmentation and feature extraction-like tasks in oceanography. For example, H. Wang et al. (2022) used a GAN to extract tidal features from snapshots of sea surface height. Similarly, Huo et al. (2024) used a multitask neural network structure to segment out mesoscale eddies and also find their edge contour from sea surface height data. In more specific applications related to oceanic tracers, Ducournau and Fablet (2016) used convolutional neural networks to upscale or increase the resolution of the sea surface temperature. Su et al. (2022) tried to reconstruct the time series of three-dimensional temperature tracer

using satellite-based surface data. They used a combination of convolution layers with a sequence architecture called long short time memory (LSTM), together popularly called ConvLSTM (Shi et al., 2015). G. Zhang et al. (2023) used a three-dimensional variant of the convolutional layer to implement a GAN model for reconstructing a three-dimensional subsurface salinity tracer field from a three-dimensional temperature profile.

PINNs or physics-informed neural networks (Raissi et al., 2019) have gained widespread popularity when applying neural networks to physical systems. PINNs allow the incorporation of conditions, such as equations, conservation laws, or other physically relevant principles during the training phase of the neural networks, making the networks much more accurate when used for scientific applications. R. Wang et al. (2020) for instance implemented PINNs by adding incompressibility constraints to their neural network loss function in order to produce turbulent flows with desirable features.

In this work we use multiple neural network models to predict tracer features from the flow. Our models are based on popular architectures of Autoencoders, UNet, and conditional GAN. Along with these, we also make use of a custom layer, the LoConv layer, that resembles a convolutional layer but without the spatial homogeneity assumption of a standard convolutional layer. Using this layer we construct a set of LoConv models. We also make use of PINNs in a mild form, where instead of adding physical constraints in the training loss function, we add a constraint on the order of magnitude of tracer spectral coefficients in the inertial range, which we call spectral loss. We test the prediction of the neural networks generating tracer physical field, spectra, flux, and other statistical features against the truth across different models, providing us insights on the features captured and missed by different models.

The paper is organized as follows. The governing equations generating the turbulent flow and tracer fields are discussed in Section 2. A detailed description of the different kinds of neural network models and their architectures are discussed in Section 3; readers not interested in specialized neural network architectures and data flow may skip this section and go directly to Section 4 that compares the computational benefits of neural networks over direct integration methods. Results from different regimes using neural networks are detailed in Section 5 along with statistics and applications. We finally conclude the study with a broader view in Section 6.

2. Flow and Tracer Dynamics Across Different Regimes

We used the two-vertical-mode model studied by Thomas and Vishnu (2022) to generate flows varying from asymptotically small to $O(1)$ Rossby numbers. The model is derived by projecting the f -plane hydrostatic Boussinesq equations onto the barotropic mode and a single high baroclinic mode, leading to just two vertical modes. Such a set up is a low dimensional representation of the oceanographic situation, where the barotropic and the first one or two baroclinic modes contain most of the balanced energy while unbalanced flow components such as near-inertial waves excited by winds blowing over the ocean or as eddies interact with bottom boundaries are high baroclinic modes (Alford et al., 2016; Wunsch, 1997). The two-vertical-mode model in nondimensional form is:

$$\frac{\partial \zeta_T}{\partial t} + Ro \hat{\mathbf{z}} \cdot \nabla \times [\mathbf{v}_T \cdot \nabla \mathbf{v}_T + \mathbf{v}_C \cdot \nabla \mathbf{v}_C + (\nabla \cdot \mathbf{v}_C) \mathbf{v}_C] = f_T - \nu \Delta^8 \zeta_T \quad (1a)$$

$$\frac{\partial \mathbf{v}_C}{\partial t} + Ro[\mathbf{v}_C \cdot \nabla \mathbf{v}_T + \mathbf{v}_T \cdot \nabla \mathbf{v}_C] + Bu \nabla p_C + \hat{\mathbf{z}} \times \mathbf{v}_C = \mathbf{f}_C - \nu \Delta^8 \mathbf{v}_C \quad (1b)$$

$$\frac{\partial p_C}{\partial t} + \nabla \cdot \mathbf{v}_C + Ro[\mathbf{v}_T \cdot \nabla \cdot p_C] = -\nu \Delta^8 p_C \quad (1c)$$

Above, subscripts “T” and “C” represent barotropic and baroclinic fields, respectively. For instance, ζ_T and \mathbf{v}_T are the barotropic vorticity and velocity vector respectively, related by $\zeta_T = \hat{\mathbf{z}} \cdot \nabla \times \mathbf{v}_T$, with $\hat{\mathbf{z}}$ being the unit vector in the z -direction. \mathbf{v}_C and p_C are the baroclinic velocity vector and pressure, respectively. ∇ is the gradient operator and $\Delta = \nabla^2$ is the Laplacian operator. The “f” terms on the right hand side represent forcing terms while the terms containing Δ^8 are hyperdissipation terms, dissipating flow fields primarily close to the grid scale, thereby preventing accumulation of energy at small-scales. The hyperdiffusivity ν is chosen to be 10^{-38} .

Equation 1 in non-dimensional form were obtained by scaling velocity with a mean flow velocity estimate, U , spatial coordinates by horizontal length scale, L , time with inertial frequency f , and pressure, p_C with fUL . Rossby

number $Ro = U/(fL)$ and Burger number $Bu = (NH/n\pi fL)^2$ are non-dimensional parameters in Equation 1, N , H , and n being the buoyancy frequency, depth of the ocean, and vertical mode number. Since the baroclinic mode is a high mode, we used a low Burger number $Bu = 0.01$ and generated flows for Ro going from 0.1 to 1.0.

We integrated Equation 1 in a doubly periodic domain $D = [0, 2\pi] \times [0, 2\pi]$ using the Fourier spectral method with RK4 scheme for time stepping. We used 768^2 grid points in space and a time step of $\Delta t = 5 \times 10^{-3}$. The barotropic vorticity was forced at low wavenumbers $k = |\mathbf{k}| \leq 5$, where $\mathbf{k} = (k_x, k_y)$ is the wavenumber vector in Fourier space, while the baroclinic mode was forced at $k = 0$, exciting pure inertial oscillations. Both barotropic flow and inertial oscillations in the baroclinic mode were randomly initialized and the forced numerical integrations were run long enough for the flow to equilibrate. The forcing scheme we used was similar to that used in Thomas and Vishnu (2022) and by varying the forcing strength we generated flows with desired baroclinic-to-barotropic energy ratios, E_C/E_T , hereafter simply called energy ratio. Thomas and Vishnu however kept the energy ratio close to 0.1 and varied Rossby number and the resulting flow structures can be seen in their Figure 1. In this work we generated a broader range of flows by varying both Rossby number and energy ratio. Specifically, we used energy ratios of 0.01, 0.05, 0.1, and 0.15 for Rossby numbers going from 0.1 to 1. For each case we calculated an effective Rossby number, $Ro_{eff} = Ro \cdot RMS(\zeta_T)$, where RMS stands for root-mean-square computed over the whole domain. This effective Rossby number for each case is given in Table 1.

The left column of Figure 1 shows snapshots of vorticity field from simulations corresponding to various pairs of Rossby number and energy ratios. In Figure 1a we have a vorticity field corresponding to $Ro = 0.3$ and $E_C/E_T = 0.01$. Here we can see coherent vortices with some amount of disruption of anticyclonic vortices. Lower Rossby number flows gave more coherent vortices with negligible small-scale features, similar to those seen in Figure 1a of Thomas and Vishnu (2022). Moving down to our Figure 1c, the Rossby number remains the same $Ro = 0.3$ as in Figure 1a, but the energy ratio is increased to $E_C/E_T = 0.15$. In this case we see an increase in small-scale structures. The coherent vortex structures are broken down further than those in panel (a), with only a few of the positive-valued coherent structures remaining. Going from Figure 1c–1e, we increase the Rossby number to 0.7, while keeping $E_C/E_T = 0.15$. We see a complete breakage of all coherent structures, with barely few sparse spots of positive-valued vortex-like structures. There is a decrease in coherent structures and increased production of small-scale structures everywhere in the vorticity field as we increase either Rossby number or energy ratio. The vorticity field for the most extreme case of $Ro = 1.0$ and $E_C/E_T = 0.15$ is provided in Figure S1a in Supporting Information S1.

Once the flows of the above nature were set up, a passive tracer, θ , was advected by the divergence-free barotropic flow and allowed to evolve until both the flow and tracer field attained statistical equilibrium. The evolution of the tracer was based on the equation:

$$\frac{\partial \theta}{\partial t} + \mathbf{v}_T \cdot \nabla \theta = f_\theta - \alpha \Delta^8 \theta \quad (2)$$

Above f_θ is a forcing term while the last term on the right-hand side of Equation 2 is the hyperdissipation term, which prevents tracer accumulation at grid scale. Here α is hyperdiffusivity, whose value is chosen to be the same as ν , the hyperdiffusivity coefficient used for vorticity.

The tracer Equation 2 was written to be consistent with the flow equations, Equation 1. However, as the tracer evolution only depends on barotropic velocity, hereafter for convenience we will drop the subscript “T” and denote \mathbf{v} for barotropic velocity and ζ for barotropic vorticity.

The tracer field was randomly initialized and the forcing we used for the tracer evolution was identical to that used by Sirohi and Thomas (2024), where the forcing maintains tracer variance at $k = 1$. The initial conditions had no role for long times with the tracer field attaining forced-dissipative equilibrium. As a result of the tracer forcing, at all times and across different regimes the tracer field at domain scale, $k = 1$, remains the same. This strategy was adopted by Sirohi and Thomas to explore the transition of tracer dispersion phenomenology as we move from mesoscales to submesoscales by generating a family of tracer fields whose tracer variance was the same at $k = 1$. This feature makes comparing tracer variance spectra of different regimes more straightforward since they all have the same value at the starting point $k = 1$, as opposed to an arbitrary stochastic forcing that may lead to different tracer variance spectra starting points.

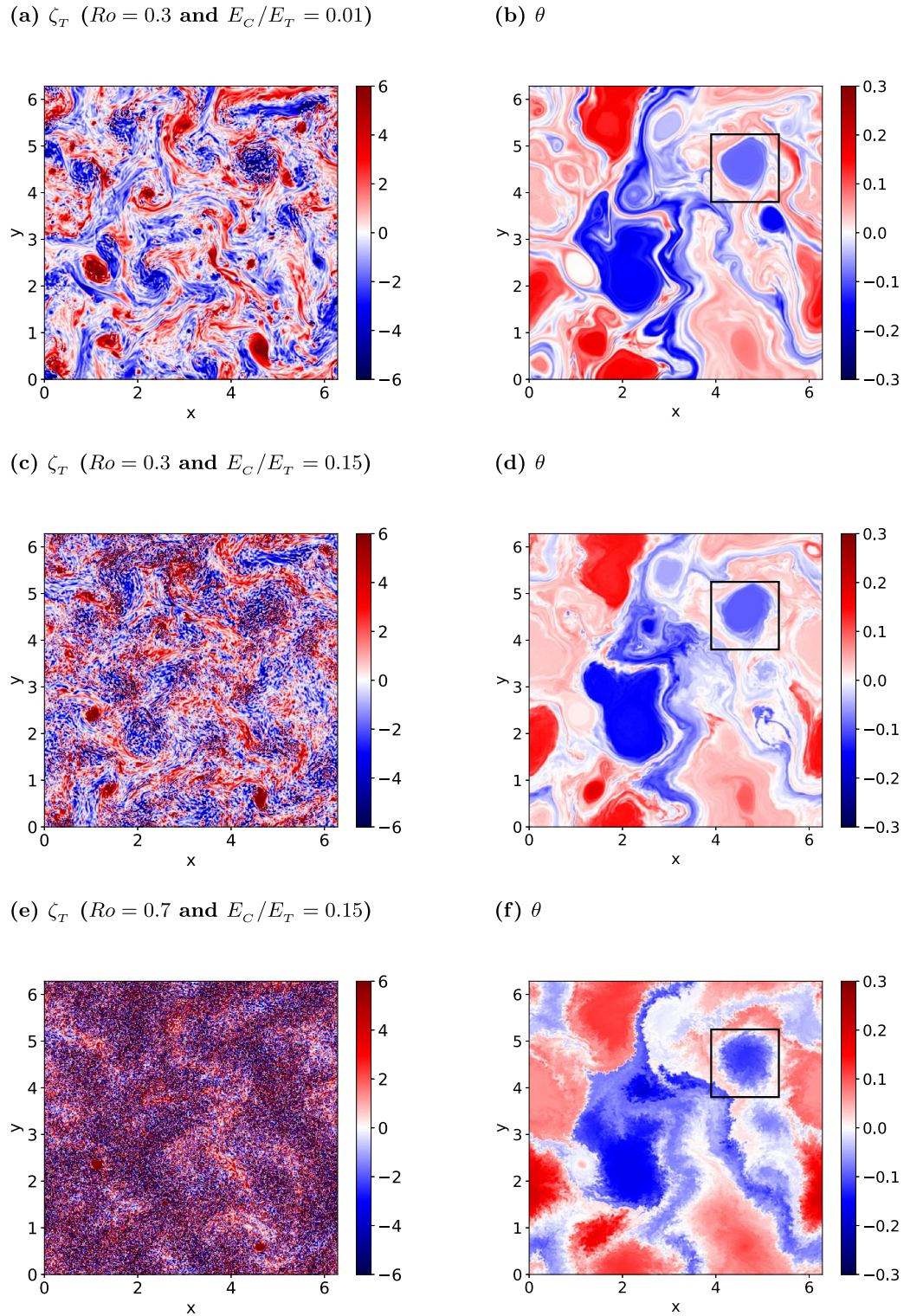


Figure 1. Physical structure of barotropic vorticity, ζ_T , and tracer, θ , for three regimes.

Of course, while the domain scale tracer variance remains the same across different regimes, the changing energy ratio and Rossby number will stir and disperse tracer very differently at scales below domain scale. Consequently, the family of tracer fields will have varying small scale structures and variance spectra inertial ranges. Our goal in

Table 1
Table Representing Effective Rossby Number, Ro_{eff} , Corresponding to Various Energy Ratios and Rossby Numbers

E_C/E_T	Ro									
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
0.01	0.13	0.33	0.62	0.90	1.28	1.89	2.43	2.94	3.46	4.09
0.05	0.13	0.38	0.78	1.12	1.66	2.32	2.98	3.70	4.42	5.07
0.1	0.13	0.43	0.83	1.44	2.22	3.10	4.09	5.04	6.07	7.07
0.15	0.14	0.46	0.95	1.67	2.78	3.83	5.03	6.34	7.61	9.06

Note. Above Ro_{eff} increases with increasing energy ratio for a fixed Ro and with increasing Ro for a fixed energy ratio. The 14 red colored cells are cases used for training neural network models while the remaining 26 are used for testing.

this work is to examine how different neural network models can predict the features of tracer structures from this family of tracer fields, given the flow, especially with an eye on small-scale or submesoscale dynamics.

We generated tracer fields in forced-dissipative equilibrium for all combinations of the 10 Rossby numbers $Ro \in \{0.1, 0.2, \dots, 1.0\}$ and four energy ratios $E_C/E_T \in \{0.01, 0.05, 0.1, 0.15\}$, as given in rows and columns of Table 1, providing us 40 regimes of flow and tracer fields. The right column in Figure 1 shows the tracer field corresponding to the vorticity field to its left. Notice that the large-scale structures of the tracer fields share similarities, this being due to the tracer forcing maintaining the tracer variance at domain-scale $k = 1$ across the regimes. Despite the similarity in large-scale structures, we see an increase in tracer dispersion and small-scale features as we move down the column. More significant changes occur along the boundaries of the tracer structures. For instance, let us focus on the coherent

structure enclosed in black boxes in panels (b), (d), and (f) of Figure 1. As we go down from (b) to (f), the enclosed blue structure in panel (b) gets increasingly stirred, and its boundary disperses out, creating more small-scale features. The plot of the tracer field corresponding to $Ro = 1.0$ and $E_C/E_T = 0.15$ is provided alongside the vorticity field in Figure S1b in Supporting Information S1.

The turbulent dispersion of tracer fields as we move from meso- to submesoscales, as seen in the right column of Figure 1, is explained in detail in Sirohi and Thomas (2024), much of the analysis being based on integrating and manipulating the tracer equation, Equation 2. In this paper, we will explore the following question: from the barotropic vorticity field can we predict passive tracer fields using neural networks? Below we will look at the different neural networks trained using data corresponding to the red cells in Table 1, so that they can take in vorticity as input and predict the tracer for the remaining untrained cases. Once again we note that the family of tracer fields we generated used a specific tracer forcing scheme and dissipative operator and the neural networks were trained using this family of tracers. Choosing a different forcing scheme or dissipation operator would require retraining the neural networks based on the corresponding tracer field family.

3. Neural Networks: Architectures and Training

To predict tracer fields from flow fields we used four types of neural network models: (a) Autoencoder, (b) UNet, (c) conditional GAN, and (d) LoConv model. We will now describe their key components and architectures, these being schematically shown in Figures 2 and 3.

3.1. Key Components of the Neural Networks

Before we discuss the architecture and pipeline of the models used, we briefly look at the building blocks of the models. All the models used in this work are constructed using one of the following: convolution based down-sampling or up-sampling blocks, plain convolution and convolution transpose layers, and our novelty, the LoConv layer. Down-sampling and up-sampling blocks are used in all our models. The down-sampling blocks, indicated by red icons in Figure 3, consists of a convolutional layer, LeakyReLU activation, and a batch normalization layer. The up-sampling blocks indicated by green icons in Figure 3, consists of a convolution transpose layer with LeakyReLU activation, and a batch normalization layer. Both blocks also have dropout layers. The function of the dropout is to randomly inhibit updating some weights during the training. This inhibition is done with probability p , which is set to $p = 0.3$ in our case. The $LeakyReLU[\alpha]$ activation with negative slope parameter α is the function $LeakyReLU[\alpha](x) = x$ when $x \geq 0$ and $LeakyReLU[\alpha](x) = \alpha x$ when $x < 0$, broadcasted over an array. We set the negative slope of LeakyReLU as 0.1 everywhere. Batch-normalization layer simply takes an array A and returns $(A - m)/v$, where m is the mean and v is the standard deviation of A (Goodfellow, 2016).

3.1.1. Convolution Layer

A convolution layer takes in image-like input in the form of an $L \times B \times C$ array where L , B , and C represent length, breadth, and number of channels in the input. It outputs an image-like array of the shape $l \times b \times c$ where $l \leq L$, $b \leq B$, and $c \leq C$ represent the output length, breadth, and number of channels, respectively. To get each

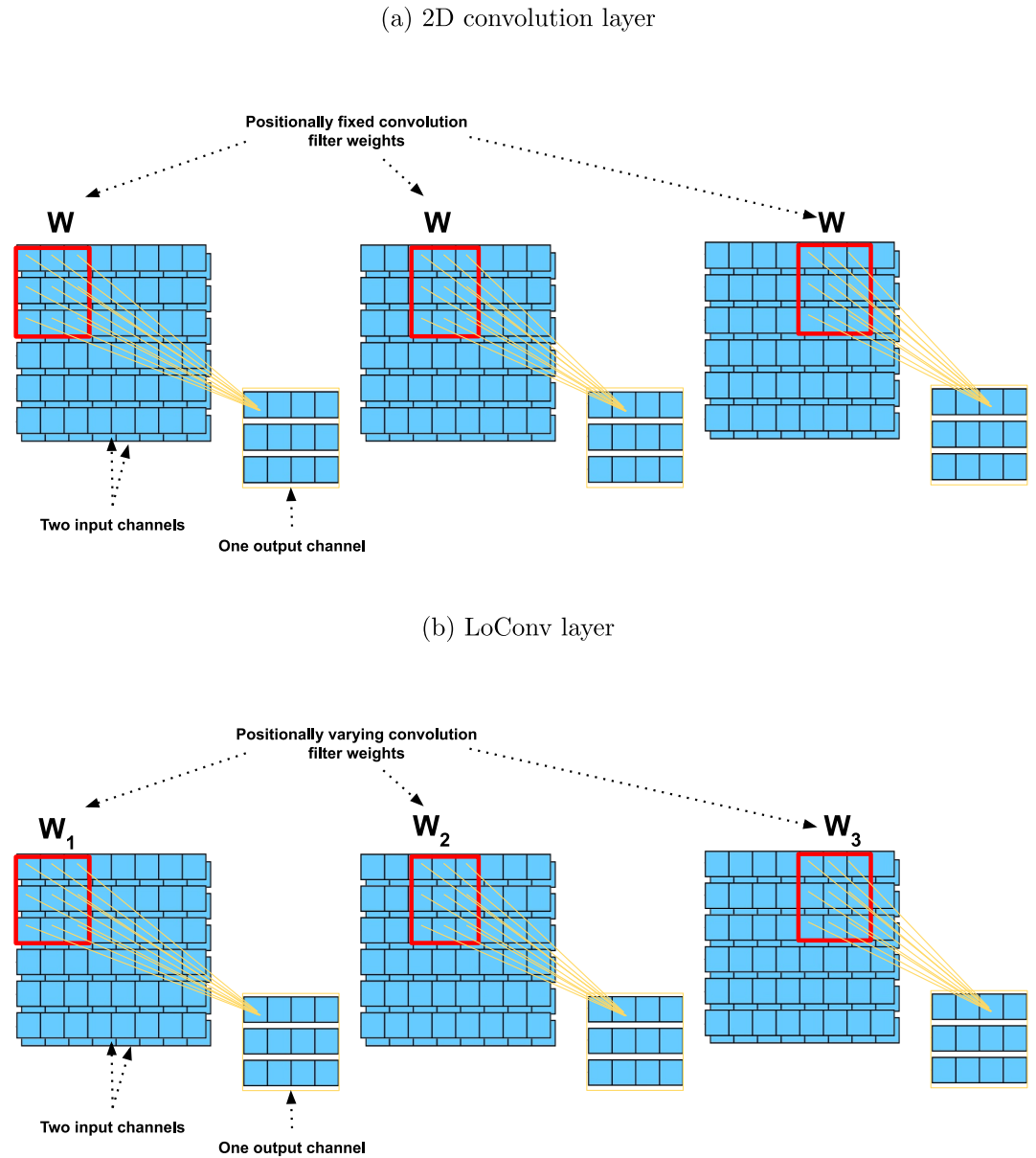


Figure 2. Mechanism of Convolution layer and LoConv layer with two input channels, one output channel, window of shape 3×3 and strides of 2×2 .

channel of the output, the convolution layer does the operation of convolution of a filter with the input image. The whole operation is determined by parameters stride (s_h, s_v) and filter shape ($n \times m$). The operation of convolution is as follows. A moving window of the same shape ($n \times m$) as the filter moves across the input image and extracts a patch of filter shape across all input channels, and the dot product is taken with filter weights, which is an array of learnable parameters of the shape $n \times m \times C$. After this the window moves to the right by s_h steps, unless steps are out of bounds for the array in that dimension, in which case the window moves down vertically by s_v steps and repeats the same. The whole operation gives a single channel of the output, and all channels are obtained in a similar manner by replacing the dot product with a matrix multiplication. Figure 2a shows the above-described mechanism of convolution for a square filter of shape 3×3 and taking strides of length two in both directions to obtain a single channel of output from 2 channels in the input. Here we note that the same weights W are multiplied across the image to get an output channel. This is in line with the assumption that features that the

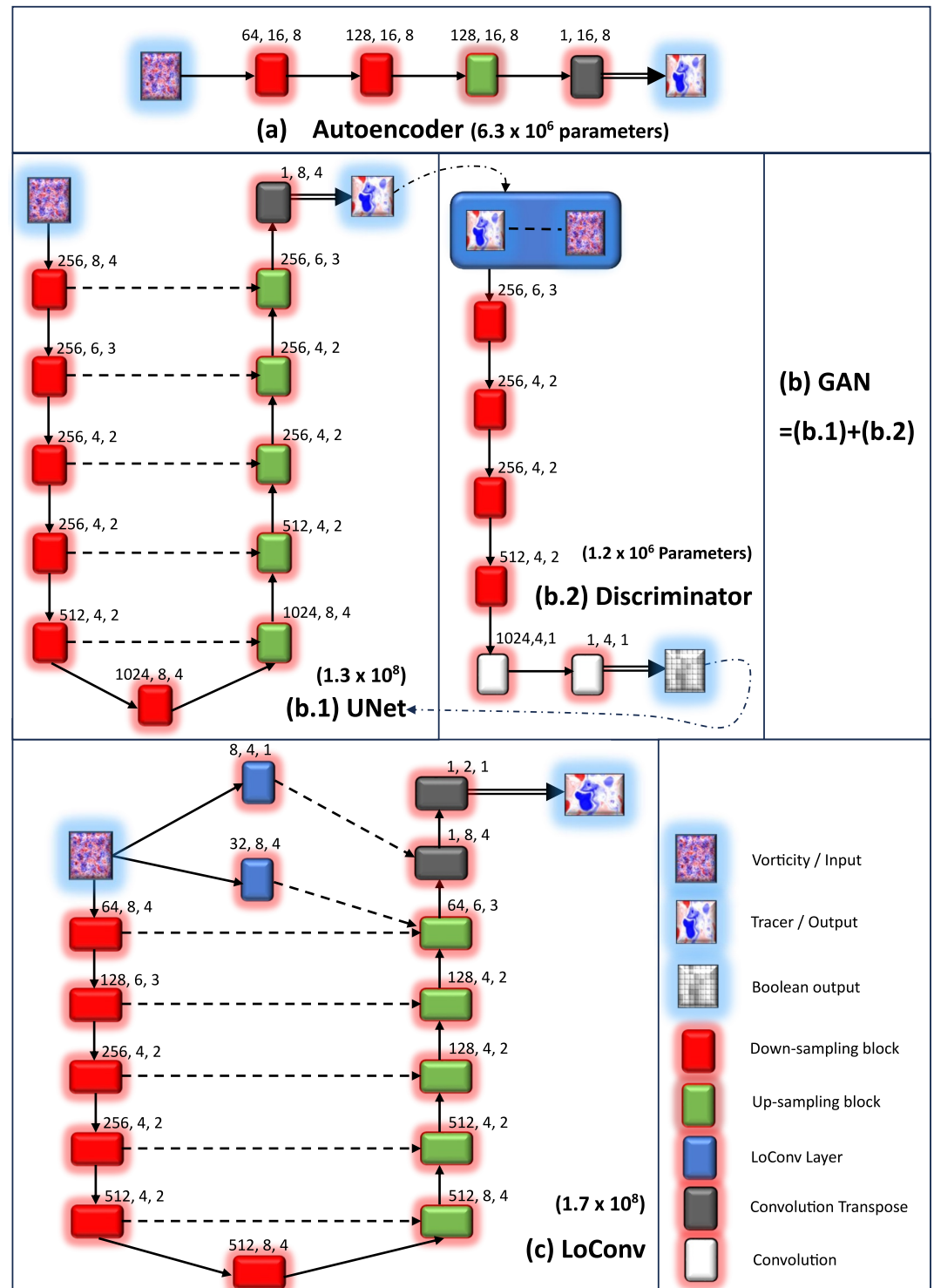


Figure 3. Illustration of the pipeline and summary of various models. The legends indicate the various layers used, and inputs and outputs. Solid lines indicate passing output of a layer to another. Dashed lines indicate concatenation along the channel (skip connection). Dash and dot lines from panel (b.1) to panel (b.2) indicates that the generated output is passed through the discriminator to get the evaluation. Dash and dot lines from panel (b.2) to panel (b.1) indicate that the discriminator output is passed to the loss function of the generator. Triad of numbers above each layer icon denotes the number of output channels, filter length, and strides to be taken in the layer, respectively. The numbers written near the model names are the total number of parameters in the neural network.

filter captures are spatially homogeneous. This is beneficial as it makes feature recognition translation invariant and also reduces the model size.

3.1.2. Convolution Transpose Layer

The whole operation of convolution to obtain an output channel can be thought of as a matrix multiplication to the flattened input array. This would be a matrix M of shape $LBC \times lbc$, obtained using the filter entries, for example, using W in case of the example in Figure 2a. A *convolutional transpose* layer takes in image-like input in the form of an $l \times b \times c$ array where l , b , and c represent length, breadth, and number of channels in the input. It outputs an image-like array of the shape $L \times B \times C$ where L , B , and C represent the output length, breadth, and number of channels. Such a transform can be obtained by multiplying matrix M^T , whose entries are either zeros (mostly) or the entries of the filter weights corresponding to matrix W . The convolution transpose layer effectively does this operation in an efficient way.

3.1.3. LoConv Layer

We saw how the operation of convolution between a filter and an image-like array gives a channel of the output of a convolution layer. Here the assumption of spatial homogeneity of features played a big role, as a fixed weight matrix corresponding to a filter was dot multiplied along the moving window across the input array. But the functional relation that we are trying to model may have some inhomogeneous elements in it.

To overcome this limitation, we use a custom layer that takes in input in the form of image-like arrays of shape $L \times B \times C$ and outputs image-like arrays of shape $l \times b \times c$, with notations the same as used in the convolution layer above. This is done by a similar operation to the convolution as shown in Figure 2a, but here the filter weight varies as we move in space along the length or breadth dimensions. Here, first, a moving window of shape $u \times v \times c$ goes across the image by strides of s , horizontally and vertically, extracting patches from the input array of the same shape. To each patch we take the inner product of a different learnable weight array. This operation on the whole image gives one channel of the output of the layer, and all channels are obtained in this manner. Figure 2b demonstrates computation of one output channel of the LoConv layer with strides of two. In the case of stride $s = 1$, the output is of the same spatial shape as the input except in the channel dimension. We note that this is similar to applying different dense layers to each of the extracted patches.

The architectures of the different neural network models are given in schematics in Figure 3. As seen from the schematics, all models we use contain down-sample and up-sample blocks. Convolution and convolution transpose layers have also been used directly as layers to extract or glue features. Convolution layers are indicated by white icons (only present in panel b.2), while convolution transpose layers are indicated by gray icons in panels (a), (b.1), and (c). We will now look at the individual model architectures.

3.2. Autoencoder

Autoencoder is popularly used for image segmentation tasks (Kingma, 2013) and is the simplest model we will use. It is also the smallest with $\sim 6.3 \times 10^6$ parameters in total, as indicated in Figure 3, near the subtitle of panel (a).

The Autoencoder consists of an input layer followed by convolution-based down-sampling blocks, decreasing the resolution of the image but increasing the number of channels or feature maps. This is followed by a convolution-transpose-based up-sampling block and a convolution-transpose layer, increasing the resolution of the image while decreasing the number of channels, until the required output shape is achieved (Goodfellow, 2016). Figure 3a shows the architecture of the Autoencoder we use. The down-sampling blocks are indicated by red rectangular icons followed by one up-sampling block, indicated by the green icon, which is followed by a convolution transpose layer indicated by a dark gray icon in Figure 3a.

3.3. UNet

UNet is a prominent deep learning image-to-image pipeline that has found widespread applications in various fields, especially in computer vision (Ronneberger et al., 2015). Its architecture is a U-shaped structure as seen in Figure 3(b.1), where the encoder part downsamples the input image to extract features, while the decoder part upsamples the features to reconstruct the target image. The key components of UNet are:

- Encoder: A series of down-sampling blocks to reduce the spatial dimensions. This part consists of the red icons seen in Figure 3(b.1). Here the first layer takes the vorticity field as input.
- Decoder: A series of up-sampling blocks to increase the spatial dimensions while gluing extracted features. This is the right side of Figure 3(b.1) with green and gray icons. The output of the final layer of the decoder is the predicted tracer.
- Skip connections: Connections between corresponding blocks of the encoder and decoder, of the same spatial dimensions. This is done by concatenating the encoder layer's output on the inputs of the decoder layers along the channel axis.

UNet's architecture when used alone is best for exact functional relations where there is no stochasticity expected in the model output or when the problem is not generative in nature, such as in image segmentation problems for example (Ronneberger et al., 2015). In our case, the total number of parameters in the UNet models were $\sim 1.3 \times 10^8$, much larger than the Autoencoder, as indicated in Figure 3.

3.4. Conditional Generative Adversarial Network

Conditional generative adversarial network is a variant of Generative Adversarial Networks that can generate or sample output, conditioned on additional information. Generally, a GAN takes in a random noise vector and outputs image-like arrays from the learned distribution (Radford, 2015). A conditional GAN (hereafter referred to simply as GAN), along with the random noise vector, also takes in a condition vector as input and outputs an image-like array from the learned conditional distribution (Gauthier, 2014). In our case, since we want to model a functional relation, the model only takes in the condition input (vorticity) and no noise.

Our GAN model consists of a generator neural network and a discriminator neural network. Figure 3 panel (b) shows the conditional GAN architecture as a combination of panels (b.1) and (b.2), corresponding to the generator and discriminator, respectively. The generator takes in the condition vector as input, which in our case is the vorticity, and outputs the tracer field. The same UNet in panel (b.1) described earlier is trained as the generator. The discriminator takes in the vorticity field, along with, either the actual tracer field or the predicted tracer field given as output of the generator as its inputs. The output of the discriminator is a vector of probabilities, that is trained to have all very high values, that is values near one, when an actual tracer is the input and all values very low, that is near zero, when a generator-predicted tracer is the input (Goodfellow, 2016). This way, it distinguishes between real and generated tracer fields. The discriminator is a composition of four downsampling and two convolution layers, as in panel (b.2). Both networks are trained simultaneously in an adversarial manner. The generator aims to produce outputs that are indistinguishable by the discriminator from real samples, while the discriminator tries to accurately classify real and fake samples. The training process involves iteratively updating the parameters of both networks based on their performance. We explain this briefly in Section 3.6 below.

3.5. LoConv

The LoConv model consists of a small UNet-like base including the red and green blocks seen in Figure 3c, but with two LoConv layers (indicated by blue icons) pushed in between. The LoConv layer, much like the convolution layer, captures local features in its input. But unlike a convolution layer, the window of weights is not spatially homogeneous, that is, there is a different window to capture the local features at each location along length and breadth, as described earlier. This helps the model capture the space-varying local dependence between the input and the target. A convolutional layer is determined by the number of output channels, filter size, and strides that the window takes. The same three parameters are also used to determine and instantiate a LoConv layer in a similar manner. But every time the filter window strides in space, a new set of weights is used to extract features at that location by taking the inner product of the weights and the input image patch. Since the model contains “local convolution-like” layers, we call this the LoConv model. The spatial inhomogeneity of LoConv is useful as tracer stirring may have spatially inhomogeneous components. In our case the LoConv has a comparable number of parameters to the UNet ($\sim 1.7 \times 10^8$).

With the model architectures described above, the data at each point get transformed as per array sizes indicated on top of each icon in the schematics in Figure 3. We refer readers to Appendix A for a detailed understanding of the data flow through the models.

3.6. Training Method

The general way of training a neural network in supervised learning is to first define a loss function that depends on the output prediction of the model, the target output, and possibly the input. Then, use some variant of the gradient descent algorithm to update the trainable weights in the model with the objective of reducing the loss function. This update is done iteratively in many steps. In each step, the models and optimizer operate over a small batch size of a few samples of the whole training data. In our case, the batch size was two. All the models were trained for 6,000 steps using the Adaptive momentum estimation (Adam) optimizer with a changing learning rate (Kingma, 2014). Learning rate (α) and moving average parameters (β) are some hyperparameters in Adam optimizer. It updates the weights W_T at time T of optimization, according to

$$W_{T+1} = W_T - \alpha m_T \quad (3)$$

$$m_0 = 0, m_T = (1 - \beta) \left[\frac{\partial(Loss)}{\partial W_T} \right] + \beta m_{T-1} \quad (4)$$

Here m_T as defined is an exponentially weighted moving average of gradients and $Loss$ is the loss function of the model whose weights are being updated.

In our case, $\beta = 0.35$ for all optimizers corresponding to each model. The learning rate hyperparameter was changed after 5,000 training steps. For the first 5,000 steps it was fixed to be $\alpha = 0.003$ and then to $\alpha = 0.0006$ for all model optimizers except for the discriminator of the conditional GAN, for which it was fixed to be 0.01.

Let ζ be the input vorticity field and θ be the target output tracer field. Denoting the predicted output tracer field by θ_{pred} , the loss function for Autoencoder, UNet, and the LoConv model is given by the L^1 norm of the difference between the prediction and target, plus a weighted spectral loss, that is

$$Loss(\theta_{pred}, \theta) = \|\theta_{pred} - \theta\|_1 + \lambda Loss_{spectral}(\theta_{pred}, \theta) \quad (5)$$

$$Loss_{spectral}(\theta_{pred}, \theta) = \log \left(\left\| \left(\left| \hat{\tilde{\theta}}_{pred} \right| - \left| \hat{\tilde{\theta}} \right| \right) \right\|_1 \right) \quad (6)$$

where L^1 norm $\| - \|_1$ for an array (a_1, a_2, \dots, a_m) is defined as $\|(a_1, a_2, \dots, a_m)\|_1 = |a_1| + |a_2| + \dots + |a_m|$. Additionally, hat denotes Fourier transform while the double-tilde denotes a filter in spectral space that retains Fourier coefficients of the tracer in wavenumber band $k \in [25, 100]$ and sets the remaining Fourier coefficients to 0.

In the loss function, Equation 5, the first part captures the difference in pointwise tracer physical structure. On the other hand, the second part given in Equation 6 with a weightage λ checks the departure of the spectrum of the tracer prediction from the true spectrum, specifically in the inertial range $k \in [25, 100]$. The upper limit of 100 was chosen to avoid the dissipation range of wavenumbers, appearing after the inertial range where our primary focus is. We did selected experiments by including the diffusive range also in spectral loss, although that had little effect on improving the inertial range spectra. In the absence of spectral loss, the predicted tracer variance spectra was seen to fall off below the actual tracer variance spectra in the inertial range, especially around $k = 25$ for different models. This led to the choice of the lower wavenumber cut off of 25 in the spectral loss function above based on a few different trials.

We will discuss results from using four different λ values: $\lambda_0 = 0$, $\lambda_1 = 1/3000$, $\lambda_2 = 1/300$, and $\lambda_3 = 1/30$. It then follows that λ_0 case has no spectral constraint while λ_3 has maximal spectral constraint enforced. To choose the values of λ we took an adhoc approach of repeatedly increasing the order of the weightage from 0 to $1/3000$, to $1/300$ to $1/30$, and observing the physical structure of the predictions. We stopped at $1/30$ when we saw that both the LoConv and UNet models showed a slight increase in the absolute error in prediction when increasing from $1/300$. The spectral loss part in the loss function can help in ensuring that the predicted spectra do not fall off much below the true spectra, as we shall see in the following section, and this approach is a mild form of PINN strategy of enforcing extra conditions in the loss function.

For the conditional GAN, the loss functions are slightly different, and the training of the generator and discriminator happens simultaneously in an adversarial manner. Let G denote the generator and D denote the discriminator. The generated tracer and the true tracer are both passed through the discriminator to get the discriminator's classification of the generated tracer $D(\zeta, \theta_{pred})$ and the discriminator's classification of the real output $D(\zeta, \theta)$. The prediction of D is a two-dimensional array of probabilities of the input being a true tracer or one predicted by the generator. The discriminator loss function is the binary cross-entropy loss function with the two classes being true tracer and predicted tracer (output of generator) (Zhang & Sabuncu, 2018). Then the generator loss, $Loss_G$, is given as

$$Loss_G(\theta_{pred}, \theta) = \|\theta_{pred} - \theta\|_1 + \lambda Loss_{spectral}(\theta_{pred}, \theta) - \gamma D(\zeta, \theta_{pred}) \quad (7)$$

From this we see that the output of the generator is used by the discriminator in its input, and the output of the discriminator is used in the generator's loss function. The term $\gamma D(\zeta, \theta_{pred})$ in Equation 7 emphasizes that minimizing the generator loss leads to maximizing the discriminator probability score for the predicted tracer, that is, “fooling” the discriminator. This is the adversarial nature of training the two networks.

All the models were trained on a Linux system with Intel Xeon CPU (2.2 GHz) 16 GB of RAM and a Tesla T4 GPU with a total memory of 15360.0 MB. The total time taken for training the models were: Autoencoder—1.17 hr, LoConv—1.517 hr, UNet—3.12 hr, and GAN—3.92 hr.

4. Time Taken by Neural Networks Versus Direct Numerical Integration

As explained earlier, it is expected to be beneficial to use neural networks to predict tracer fields directly from the flow, instead of numerically integrating the tracer equation along with the flow. To evaluate the benefits of using neural networks, we will now quantify the computational time taken by the two processes.

To compare the two processes on equal footing, we will compare the neural network inference time against the time for numerical advection of the tracer equation on the same hardware configuration. Specifically, we chose a broadly accessible general-purpose laptop configuration of an Intel Xeon CPU (2.20 GHz), 16 GB RAM and no GPUs. Consider the process of generating tracer snapshots at 20 time instances separated by a unit time interval $t_0, t_0 + 1, \dots, t_0 + 19$. The separation of unit time intervals is useful since generating tracer statistics such as averaged spectra, flux, correlations, etc. often require averaging over different tracer field instances and often a unit time interval is the preferred time gap that allows efficient time averaging. Of course, this time gap can be higher than unity as well, depending on specific averaging requirements.

For generating tracer fields from the neural networks we collected the 20 vorticity snapshots corresponding to the same time instances by integrating the flow equations. With this information, the neural network models took the following time to generate the 20 tracer fields: Autoencoder—4.17 s, LoConv—39.89 s, UNet and GAN—64.67 s. On the other hand, direct numerical integration of the tracer equation needed a time step of 0.005 for stable integration, leading to 200 time steps in a unit time interval. This numerical integration approach took 11261.12 s or 3.13 hr for generating the 20 tracer fields, significantly more than the neural network prediction time. Note that time for direct integration assumes that we are starting with a tracer field that is already in forced-dissipative equilibrium. The direct integration time can be much longer if we include the transient evolution time of the tracer field until it reaches equilibrium. Both processes, neural network predictions and direct numerical integration was seen to take proportionally more time for generating more frames, such as 40 frames instead of 20. In general, the slowest neural network was faster than direct integrations by a factor of 174. This advantage will improve further if the time gap between predictions were to be more than unity. Also note that the time taken for the two processes will proportionally reduce if we scale up the hardware, such as using higher configuration CPUs or GPUs, since both processes will benefit proportionally.

Recall from the previous discussion that the Autoencoder, LoConv, UNet, and GAN took 1.17, 1.517, 3.12, and 3.92 hr, respectively for training on a 16 GB GPU system. The above estimates compares the two processes on the same hardware but ignores the time taken for training neural networks. Notice that the GAN model takes more time in training alone (3.92 hr) than the direct integration process (3.13 hr). Therefore, on a quick look, predicting a discrete sample of 20 frames using neural networks does not seem to be much beneficial when compared to direct integration if we factor in the training time. However, neural networks can still be beneficial when

predicting a batch of tracer fields corresponding to different regimes. To quantify this, recall that the 14 red cells in Table 1 were regimes used for training while the 26 white cells are regimes used for predictions. Consider neural networks generating 20 tracer fields for each of these 26 regimes. These predictions will need 26×4.17 seconds = 1.8 minutes for the Autoencoder, 26×64.67 seconds = 28 minutes for the UNet and GAN, and 26×39.89 seconds = 17.3 minutes for the LoConv. The total time taken by the different neural network models to predict 20 tracer fields across 26 regimes, obtained by summing the training and prediction time, are: Autoencoder - $1.17 + 1.8/60 = 1.2$ hr, LoConv— $1.517 + 17.3/60 = 1.80$ hr, UNet— $3.12 + 28/60 = 3.59$ hr, GAN— $3.92 + 28/60 = 4.39$ hr. In contrast, direct integration for generating 20 tracer field snapshots across 26 regimes would take 81.33 hr, a factor of almost 20 more than GAN, the slowest neural network. This calculation again assumes that we are starting direct integration from an equilibrated tracer field. The total time taken for direct integration will be much longer if we factor in the time taken by the tracer field to reach forced-dissipative equilibrium.

The above discussion informs us that comparing advantages of neural networks over direct integration does not often lead to black and white answers. If we discard the training time, neural networks without doubt are significantly faster than direct integration. On the other hand, if we factor in the training time, neural networks can be inefficient in saving computing time if the predictions are sought for a single regime. However, neural networks can be beneficial when predicting tracer structures for a broad set of regimes, that is a bigger sample, even with training time factored in. In such cases, direct integration takes a lot more time to generate a large set of samples than the total time taken by neural networks.

5. Neural Network Predictions

As explained earlier, our data consists of 40 regimes of flows, corresponding to each pair of Rossby number and energy ratio as highlighted in Table 1. We chose 14 of these 40 regimes randomly and used only data corresponding to those 14 regimes to train all our models, these being indicated by red color in Table 1. Only 35% of total data was therefore used for training and the remaining were used for evaluating the predictions ahead. For each regime, we collected 40 snapshots of tracer and vorticity for training, implying that our training data consisted of $40 \times 14 = 560$ snapshots. In this section we will compare and contrast the tracer predictions made by the Autoencoder, LoConv, UNet, and GAN against the true tracer field. Each of these 4 architectures were trained with 4 different loss functions. The loss functions being parameterized by varying the weightages λ of spectral loss in Equations 5 and 7, discussed earlier. The four weightages λ used for training the models are $\lambda_0 = 0$, $\lambda_1 = 1/3000$, $\lambda_2 = 1/300$, and $\lambda_3 = 1/30$. Consequently, we have 16 models in total, one for each combination of architectures and weightages of spectral loss. We will indicate a model trained with weightage λ as Model- λ . For example, a combination of UNet trained without any spectral loss term will be indicated as UNet- λ_0 . We will systematically compare the predictions of all of them by examining the predicted tracer physical structures, tracer variance spectra, and flux.

5.1. Overview of Prediction Errors

All 16 models take in the vorticity field as input and output the predicted tracer field. The mean of the percentage of absolute error for each of the models, where the mean is taken over prediction snapshots across the regimes, is shown in Figure 4a. The mean percentage absolute error of n predicted samples is defined as

$$\mathcal{E}(\{Y_i^*\}, \{Y_i\}) = \frac{1}{n} \sum_{i=1}^n (\|Y_i^* - Y_i\|_1 / \|Y_i\|_1) \times 100$$

Here Y_i^* is the prediction and Y_i is the truth. We see that the LoConv model with any weightage of spectral loss outperforms all other models in terms of absolute error. The error for the LoConv model ranges between 12.45% and 14.25%, and is seen to slightly increase with increasing weightage of spectral loss. The UNet models overall ranks second, with the value of the error ranging from a minimum of 16.86% for $\lambda = \lambda_2$ to a maximum of 25.96% for $\lambda = \lambda_1$. The GAN models have an error ranging from a minimum of 24.50% for $\lambda = \lambda_2$ to a maximum of 36.86% for $\lambda = \lambda_3$. Autoencoder has the worst performance in terms of absolute percentage error, with error decreasing with increasing weightage of spectral loss, ranging from 34.65% to 42.46%. The comparatively poor

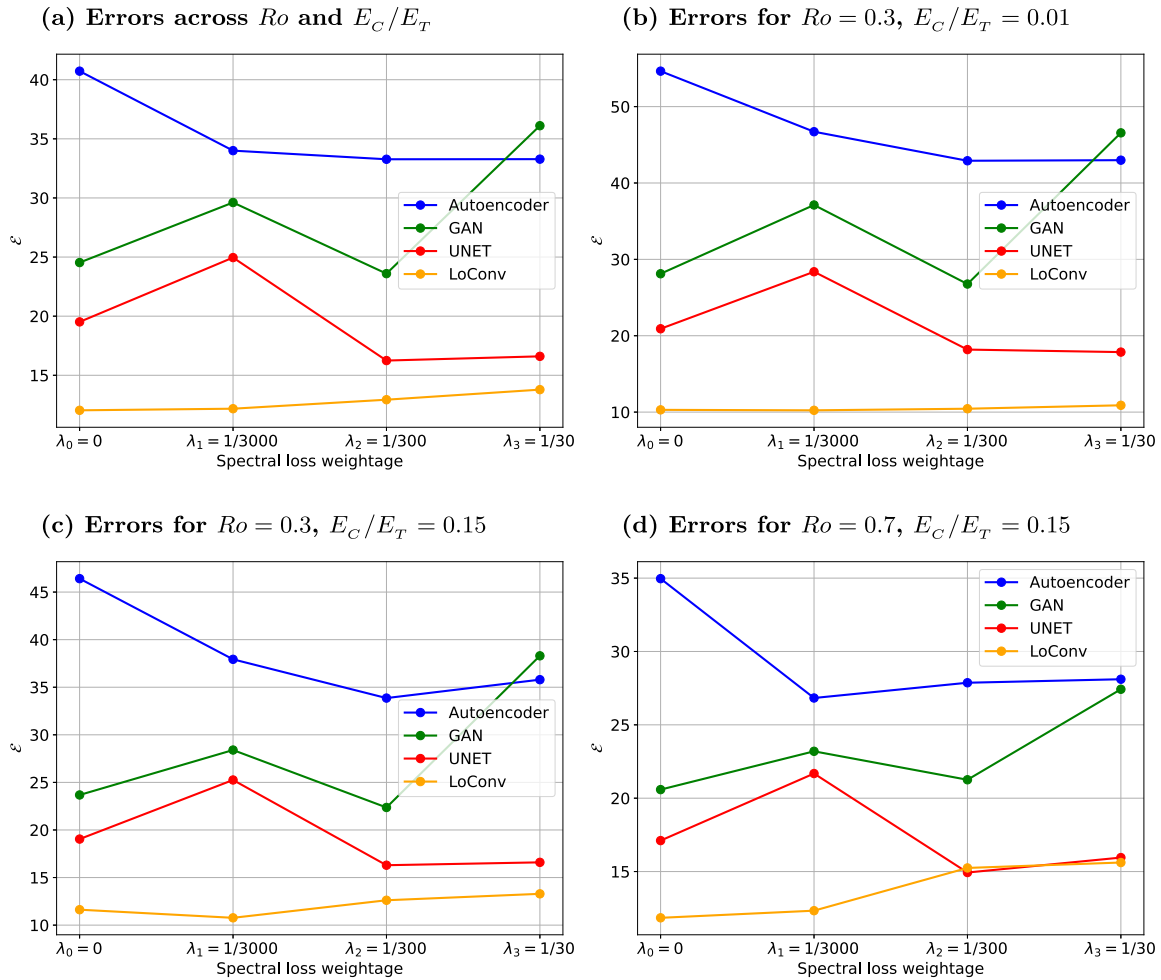


Figure 4. Error comparison of the models across different weightages of spectral loss. Each figure presents the mean percentage absolute error for all 16 models. Four different colors are used to represent the four architectures and the four points on the x -axis represent the four weightages for spectral loss.

performance of Autoencoder is expected, as the Autoencoder model is very small, with number of parameters approximately 6×10^6 compared to the other models whose number of parameters is of the order of 10^8 .

Figures 4b–4d show errors corresponding to the three regimes shown in Figure 1. Figure 4b is an instance for $Ro = 0.3$ and energy ratio 0.01, and Figure 4c is an instance for the same Rossby number $Ro = 0.3$ but with a higher energy ratio 0.15. The trend of errors across models in these two specific cases is the same as the trend in the average error plot in Figure 4a. Figure 4d shows errors corresponding to $Ro = 0.7$ and $E_C/E_T = 0.15$, this being a case of strong stirring of the tracer field. In this case we see that the error for LoConv and UNet models is comparable when they are trained with a high weightage of spectral loss. This can be seen in Figure 4d where the orange line and red line overlap for λ_2 and λ_3 . The error plot for the most extreme case of $Ro = 1.0$ and $E_C/E_T = 0.15$ is provided in the Figure S2 in Supporting Information S1. In that case, UNet- λ_2 or UNet- λ_3 have errors around 16%, and corresponding LoConv models have errors around 20%.

From the point-wise absolute error, which is a useful metric for evaluating model predictions, LoConv seems to do very well, while UNet seems to match with LoConv at high Rossby numbers for spectral weightages λ_2 and λ_3 . Below we will compare the models' predictions further based on physical plots and other statistics.

5.2. Tracer Physical Structure and Spectrum

Here we will look at the tracer physical structure predictions from different models. Figures 5–8 show comparative physical structures of the predicted tracer fields. These are instantaneous predictions from randomly

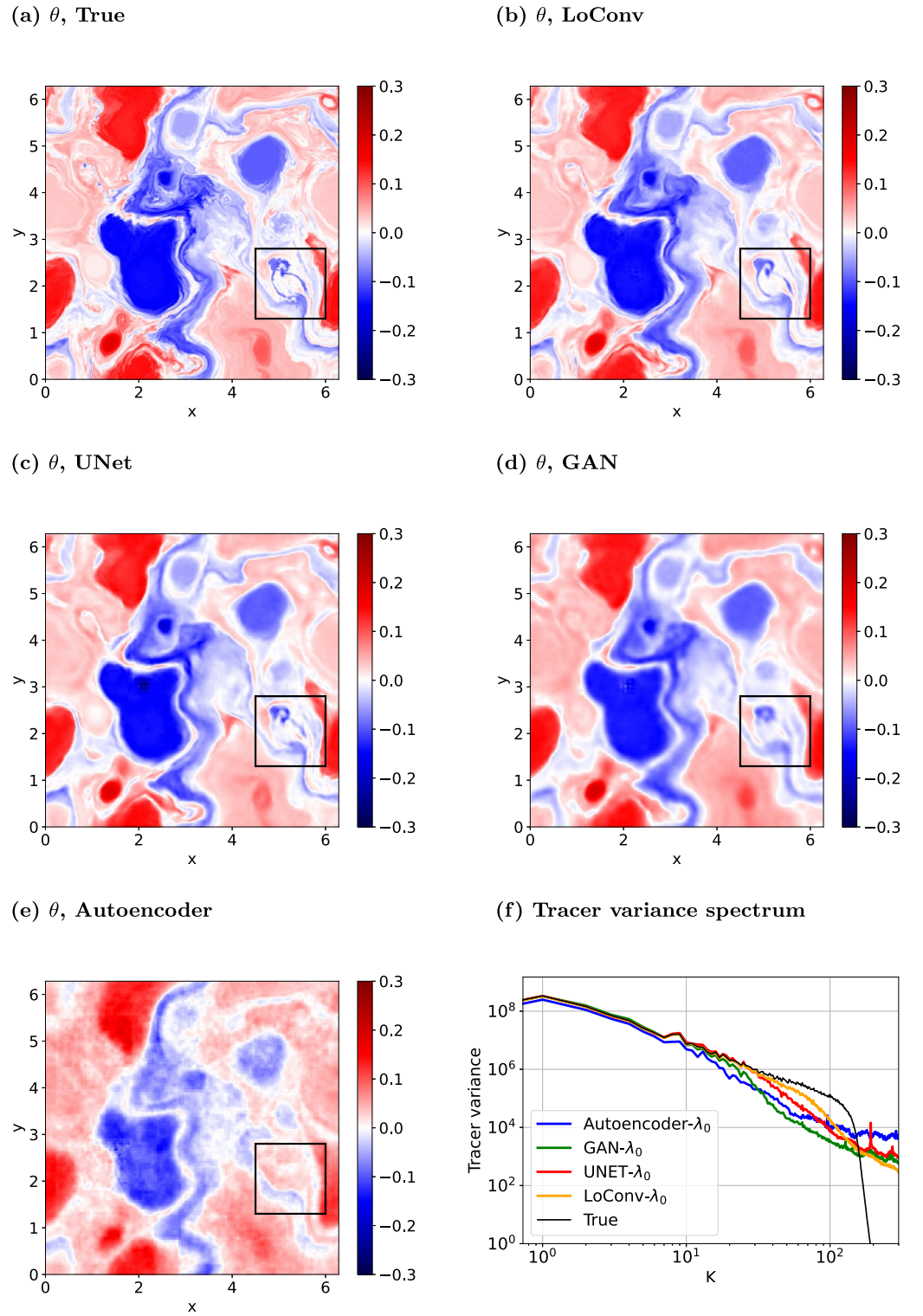


Figure 5. Comparison of predictions of tracer physical structure and variance spectrum from models with $\lambda = \lambda_0$ for $Ro = 0.3$ and $E_C/E_T = 0.15$. The black boxes highlight a specific region for comparison.

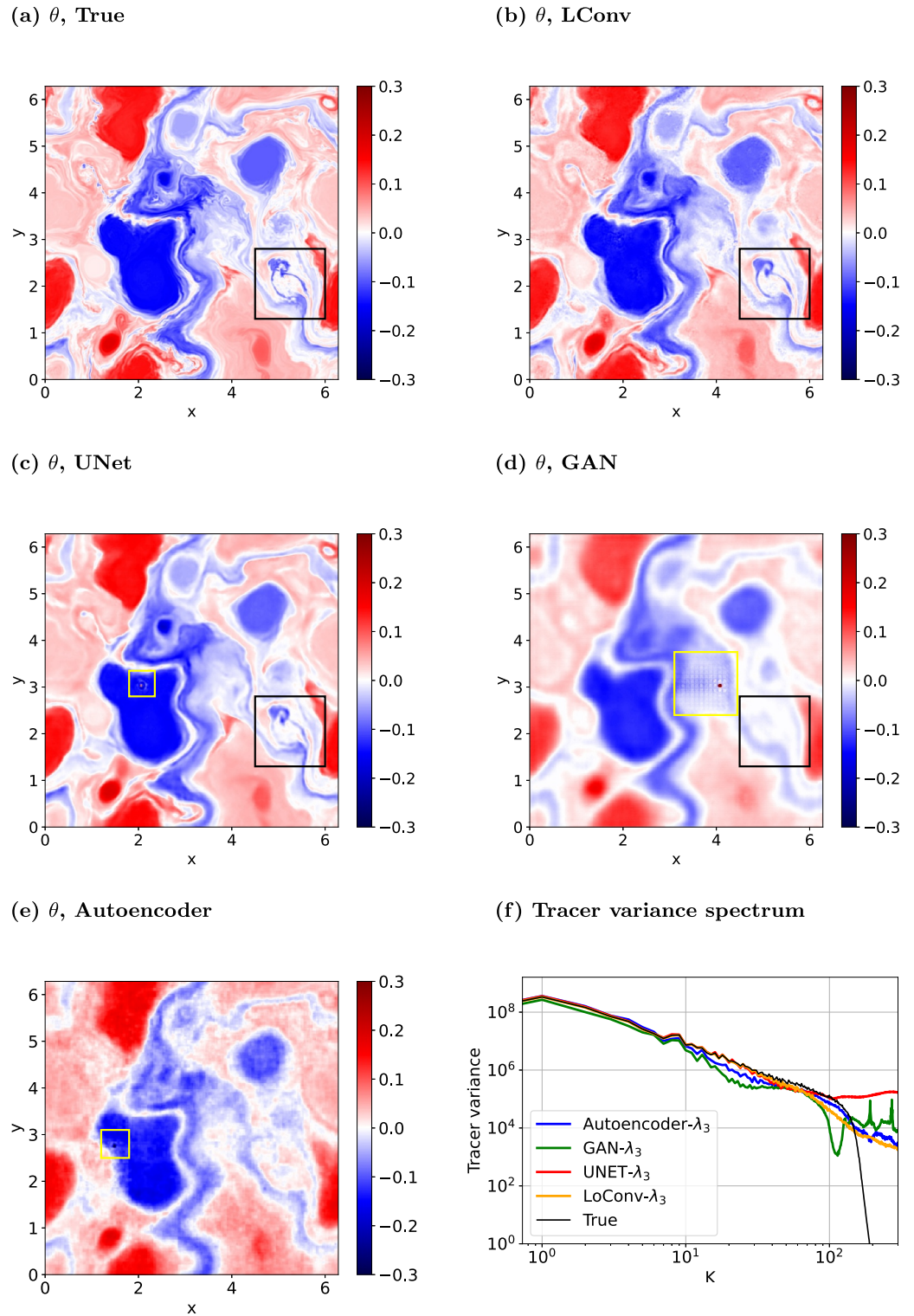


Figure 6. Comparison of predictions of tracer physical structure and variance spectrum from models with $\lambda = \lambda_3$ for $Ro = 0.3$ and $E_C/E_T = 0.15$. The black boxes highlight a specific region for comparison. Yellow boxes contain a region with anomalous physical features.

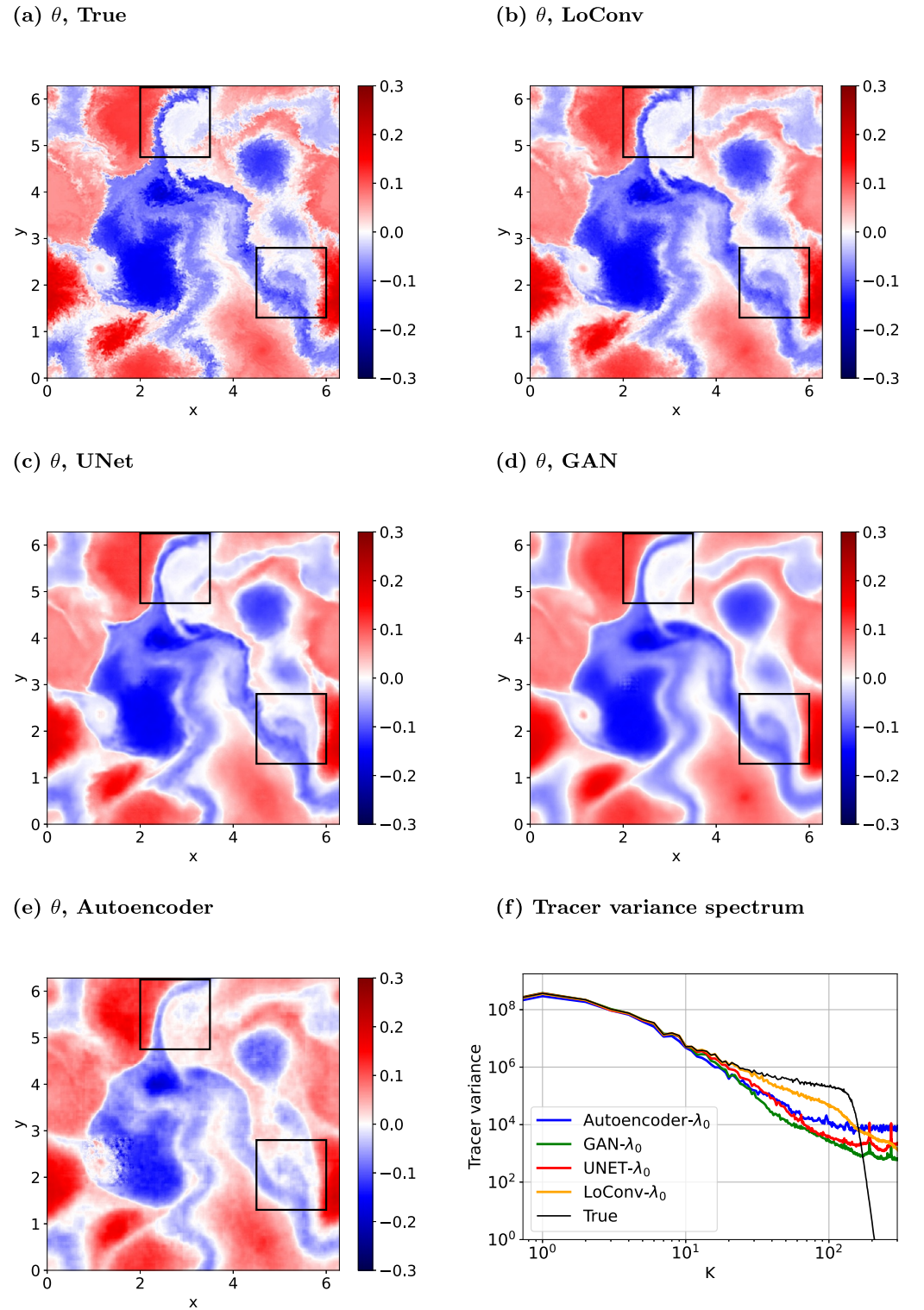


Figure 7. Comparison of predictions of tracer physical structure and variance spectrum from models with $\lambda = \lambda_0$ for $Ro = 0.7$ and $E_C/E_T = 0.15$. The black boxes highlight specific regions for comparison.

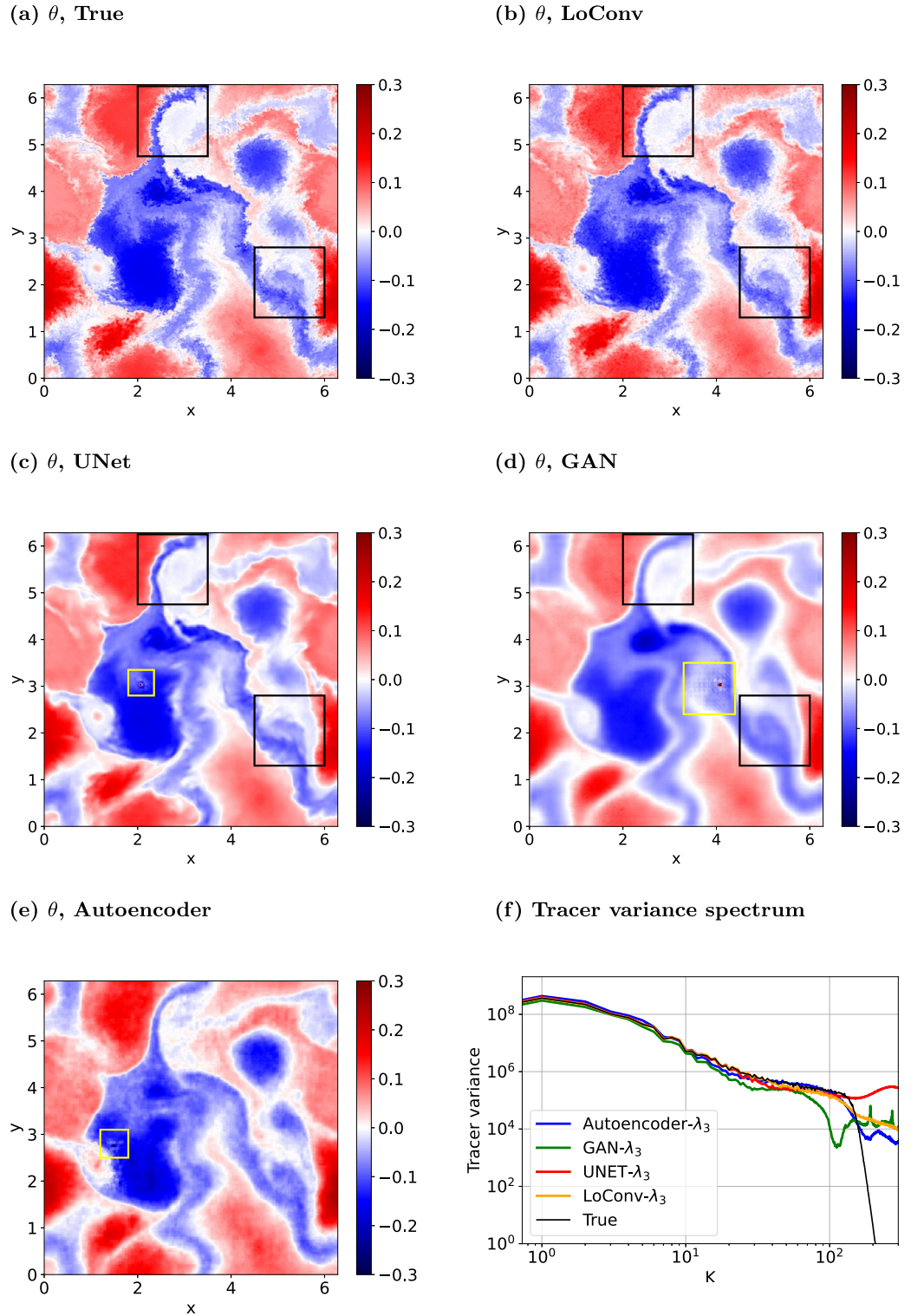


Figure 8. Comparison of predictions of tracer physical structure and variance spectrum from models with $\lambda = \lambda_3$ for $Ro = 0.7$ and $E_C/E_T = 0.15$. The black boxes highlight specific regions for comparison. Yellow boxes contain a region with anomalous physical features.

chosen times from two pairs of energy ratios E_C/E_T and Rossby numbers Ro . In Figures 5 and 6 we see predictions for $Ro = 0.3$ and $E_C/E_T = 0.15$, corresponding to all model architectures, but in Figure 5, no spectral loss ($\lambda = \lambda_0$) is used, whereas in Figure 6, maximum spectral loss of $\lambda = \lambda_3$ is used. Similarly in Figures 7 and 8, we see predictions for data from $Ro = 0.7$ and $E_C/E_T = 0.15$, corresponding to all model architectures, but in Figure 7, no spectral loss is used, whereas in Figure 8, maximum spectral loss of $\lambda = \lambda_3$ is used.

Let us do a qualitative comparison of various models' predictions for $Ro = 0.3$ and $E_C/E_T = 0.15$. First we will compare models trained without any spectral loss, that is, $\lambda = \lambda_0 = 0$. The predicted tracer fields in this case are shown in Figure 5. All the models except Autoencoder- λ_0 seem to produce the large coherent structures in the true tracer field in Figure 5a accurately. The Autoencoder produces artificial patches of low tracer value within the large structures in Figure 5e. Let us now focus on the region inside the black box near the bottom right region of Figure 5a of the true tracer field obtained numerically. This region has some sharp small-scale features, such as two prominent blue streaks, generated by flow stirring. We will use this region to compare different predictions, although the reader may infer qualitatively similar results using any other specific region in the domain. We note in Figure 5b that the LoConv- λ_0 model prediction captures a lot of small-scale features and produces features similar to the true field, such as the thin blue streaks in the black box. The UNet- λ_0 does not seem to produce such minute details, and the predictions in Figure 5c also look diffused compared to the true tracer field or the LoConv- λ_0 prediction. One of the streaks present in the box region is completely lost in the UNet prediction. The GAN- λ_0 prediction in Figure 5d seems even more diffused than the UNet and misses most small-scale features. For instance, both blue streaks in the boxed region are lost. The Autoencoder- λ_0 predictions in Figure 5e are way off. Again looking at the boxed region, none of the thin blue streaks present in the true tracer field seem to be present in the prediction.

Figure 5f shows the spectrum of tracer variance with a black line indicating the true spectrum. The spectrum of Autoencoder- λ_0 prediction, shown in blue color, has much lower variance across all scales. GAN- λ_0 predicted spectrum starts dropping at around wavenumber 20. UNet- λ_0 spectrum shown in red and LoConv- λ_0 spectrum shown in orange start to drop at around wavenumber 30, although the drop from the true spectrum is far more for UNet- λ_0 than the LoConv- λ_0 prediction. This explains the more diffusive nature of predictions by UNet- λ_0 or GAN- λ_0 . In the dissipation range, above wavenumber $k = 100$, the true spectrum in the black curve drops steeply due to the usage of hyperdissipation in the tracer Equation 2. While all the predictions drops gradually, the steep drop in the true spectra, which is characteristic of the hyperdissipation range, is absent in the predicted spectra. This is expected since there is no natural diffusion in spectral space applied to the neural network models, and the learning is based on pattern recognition from the data in physical space.

In principle we could enforce an extra diffusion criterion at high wavenumbers, something we were able to implement successfully to get faster dropping spectra predictions in the dissipation range. However, it is important to note that hyperdissipation used in the tracer Equation 2 is an artificial means to get an extended inertial range with a minimal dissipation range; this technique giving us almost two decades of inertial range. In reality, tracer inertial ranges can span several orders of magnitude, compared to the two decades we obtain in our data sets. Consequently, the neural net predictions having an inertial range that matches with that of the true spectra with a gradual decay after the inertial range is more realistic and therefore we do not enforce any extra constraints to diffuse tracer fields in the dissipation range during the neural nets' training.

In Figure 5f notice that the predictions' spectra drops such that the inertial range is not well captured by the predictions. To address this we incorporated spectral loss function, explained with Equations 5 and 6. Gradually increasing the weightage λ in Equation 5, we found the predictions to have inertial range that resembled that of the true tracer field spectra. Figure 6 shows predictions for the same Rossby number 0.3 and energy ratio 0.15 but from models that have been trained with maximum weightage of spectral loss, that is, $\lambda = \lambda_3$. We note from Figure 6f that the spectrum of all the predictions improves drastically in the inertial range when comparing it with the spectra in Figure 5f.

We can again focus on the boxed regions in Figures 6a–6e for cross-comparisons. In panel (b) we note that LoConv- λ_3 has a slight increase in very small-scale features, but overall there is negligible improvement in small-scale feature production. This is expected as LoConv predictions were already accurate, and there was little room for improvement. The improvement in spectrum for the UNet- λ_3 model slightly reflects in the physical structure in panel (c). The two blue streaks in the boxed region are more prominent compared to the UNet- λ_0 prediction in Figure 5c. But the image still looks quite smoothed out, with a generic lack of small-scale features. Additionally, a

sharp dot-like structure appears near the center of the physical plot, shown in a yellow box in Figure 6c. The structure takes a large positive value in a region with a high negative value, with tracer value fluctuating around it concentrically. We found these kinds of anomalous dot-like structure generation to be common to most of the tracer predictions from UNet, GAN, and Autoencoder models when trained with spectral loss. The introduction of such artificial structures can change the Fourier coefficients of the predicted tracer appropriately to match the spectrum of true tracer while deteriorating physical space features as a side effect.

Despite introducing spectral loss, the GAN- λ_3 prediction in Figure 6d does not seem to have improved. When compared to Figure 5d, the GAN- λ_3 looks more diffused. The spectrum of GAN- λ_3 prediction in green in Figure 5f falls slightly at very large scales, reflecting a lighter-looking tracer field. Again, an anomalous sharp dot-like structure appears, this being highlighted with the yellow box in Figure 6d. The tracer value fluctuates around it concentrically. We also see a checkerboard pattern in the yellow box. We found such checkerboard-type patterns to be typical for GAN models' tracer predictions and should be a consequence of adding a discriminator to the model training. The Autoencoder- λ_3 prediction in Figure 6e show improvement compared to Autoencoder- λ_0 prediction in Figure 5e. First, the large-scale features are improved as the low-intensity patches in coherent structures have reduced. There are a lot of small-scale features appearing in the predictions, although as a downside, the small-scale features seem arbitrary and do not reflect realistic flow structures. Additionally, while not so prominent as in the UNet or GAN, a less sharp version of the dot structure also appears in the prediction of Autoencoder when trained with high spectral loss. The yellow box in panel (e) of Figure 6 highlights this artificial structure in the prediction.

We saw a consistent improvement in the spectrum in the inertial range for all models, seen by comparing panel (f) of Figures 5 and 6. In the dissipation range seen in Figure 6f, beyond wavenumber 100, we see that the UNet spectrum overshoots the truth while the GAN spectrum has large oscillations. The Autoencoder and LoConv predictions gradually decay. Overall, we see that spectral loss seems to introduce more artificial features in the UNet and GAN predictions.

We next look at predictions for the case $Ro = 0.7$ and $E_C/E_T = 0.15$. First we look at all models trained with no spectral loss, that is, $\lambda = \lambda_0 = 0$, in Figure 7. Notice that the true tracer field in panel (a) is much more dispersed than the lower Rossby number tracer field seen before. We have again drawn black boxes to highlight two regions. One near the top boundary, where we have a long thin blue stream of tracer field and another in the bottom right, where we see a thicker blue stream of tracer along with a blue cloud of small-scale dispersive structure. The LoConv- λ_0 prediction in Figure 7b accurately produces the large structures and some small features such as the ones in the box on the bottom right. But the small-scale dispersive structures seem slightly diffused, as seen in the top box region. The prediction of UNet- λ_0 in panel (c) is much more diffused. The boundary of the blue stream in the top box is very smooth, and the cloud in the bottom box near the right boundary is also diffused. The GAN- λ_0 prediction in panel (d) completely misses all small-scale structures and the predictions look more diffused than UNet predictions, easily seen by looking inside the box regions. The Autoencoder- λ_0 prediction in panel (e) again creates patches of low absolute value of tracer. They produce more small-scale features than the GAN- λ_0 , but the features don't reflect the dispersive nature of the true tracer field.

In general we see that the predictions in Figure 7 are diffused or smoother than the true tracer field. This is reflected in the variance spectrum of the predictions. In panel (f) of Figure 7, the spectrum of all models other than LoConv- λ_0 drops off and diverges from the true tracer variance after the first 10 wavenumbers. The spectrum of LoConv- λ_0 starts to diverge after the first 30 wavenumbers and does not drop off by much even after that.

The predictions for the same Rossby number and energy ratio, but from models trained with the highest weightage of spectral loss, $\lambda = \lambda_3$, are shown in Figure 8. From panel (f) we see that the spectrum of all the models other than GAN is accurate for two decades of wavenumbers and only diverges near the dissipation range. This greatly improves the visual quality of LoConv prediction. The LoConv- λ_3 prediction in Figure 8b captures all the large structures accurately and also produces dispersive small-scale features that look similar to the ones in the true tracer field. The boundaries of the large structures are better captured in Figure 8b than in Figure 7b, as seen by looking at the black boxed region at the top boundary. The small-scale structures in the cloudy region in the box in the bottom right also look dispersed, similar to the true tracer. On the other hand, the UNet and GAN predictions in Figure 8 does not seem to have much improvement compared to the ones in Figure 7; they both look very diffused in comparison to the true field. The Autoencoder prediction seem to have no improvement. Additionally, we again see a dot-like structure with tracer signs fluctuating concentrically around it, in the UNet and GAN

prediction. A dull version of the dot-like structure is present in the Autoencoder prediction as well. The dot structures have been indicated by yellow rectangular boxes for UNet and Autoencoder in panel (c) and (e) of Figure 8 respectively. For GAN prediction, the dot structure can be found inside the bigger yellow box in panel (d). This larger yellow box also contains a checkerboard structure that was not present in the true tracer. Recall that we observed a similar pattern in GAN- λ_3 predictions for Rossby number 0.3 and energy ratio 0.15 earlier.

Despite the improved spectrum in inertial range, the dissipation range varies quite a bit between models. We see that the UNet spectrum in red increases in the dissipation range while the GAN spectrum oscillates. Both LoConv and Autoencoder spectra drop gradually in the dissipation range, with the LoConv spectrum being the smoother one.

The major qualitative inferences made with the cases discussed above holds for all other cases for which tracer predictions were made. Since it would consume too much space, we have provided some additional comparisons in the SI. The predictions for $Ro = 0.3$ and $E_C/E_T = 0.01$, whose tracer field was shown in the top row of Figure 1, are shown in Figures S3 and S4 in Supporting Information S1, corresponding to predictions from models trained with no spectral loss and maximum spectral loss, respectively. These figures also contain comparisons of their spectra. The Supporting Information S1 also contains the predictions for the most extreme case of $Ro = 1$ and $E_C/E_T = 0.15$, from all the models, along with a comparison of spectra. These are provided in Figures S5 and S6 in Supporting Information S1, corresponding to models trained with no spectral loss and maximum spectral loss, respectively. The reader will find that the main features discussed above can be seen in each of the cases shown in the Supporting Information S1.

5.3. Effect of Spectral Loss on Predictions

Neural network models trained to generate image-like objects by learning to reduce L^p loss are prone to a spectral bias towards larger wavenumbers, as shown in Rahaman et al. (2019). This causes the smoothening of predictions for a lot of models, as we have seen in the above discussion, and is a natural hindrance in problems where small scales are important. To sort this we gave some weightage to accurately predict the Fourier coefficients for wavenumbers in the inertial range, implemented by adding a weighted spectral loss in the training loss; recall Equations 5 and 6. Incorporating spectral loss was seen to improve the predictions, as seen by comparing Figures 7 and 8. The LoConv model improved with this approach and there is appropriate production of small-scale features. Nevertheless, this in turn increases the absolute error as seen in Figure 4. In fact, there are several instances at high Rossby numbers where the UNet with large weightage for spectral loss ($\lambda = \lambda_2$ or λ_3) has comparable or even lower absolute error in predictions than that of the LoConv model: recall Figure 4d and Figure S2 in Supporting Information S1.

Despite UNet predictions having comparable or lower net error than LoConv for certain higher Rossby numbers, the latter's predictions were consistently much better than the former. To see this, let us take the example in Figure 8, which has predictions for $Ro = 0.7$ and $E_C/E_T = 0.15$, and models are trained with a maximum weightage of spectral loss λ_3 . In this example the absolute error of both the UNet- λ_3 and the LoConv- λ_3 are approximately 16%, as seen in Figure 4d. But the UNet- λ_3 prediction in Figure 8c looks like a very smoothed-out version of the true tracer field and lacks the dispersion seen in the true tracer field in Figure 8a. This can especially be seen along the boundaries of the large tracer structures, where fine scale features are missed in the prediction. This is not the case with the LoConv model, as the predictions seem to produce fine-scale dispersive structures. While these structures generated as a result of enforcing spectral loss are qualitatively correct, they may not necessarily align pointwise precisely with those of the actual tracer field and this can cause an increase in absolute error.

Recall that we added a weighted spectral loss in the training loss of the model with the expectation that there will be a better match in the Fourier coefficients in the inertial range, and hence the production of better small-scale features. Since the model is trained to minimize the spectral loss along with the absolute loss, we expect a slight increase in absolute errors, as minimizing one loss has to be compensated with the other. In return we expect to get better quality of dispersion in the predictions. Figure 9 shows the spectra of predictions from all models compared with the true spectrum for $Ro = 0.7$ and $E_C/E_T = 0.15$. Notice that the spectrum of prediction from each architecture converges to the spectrum of the true tracer field in the inertial range as we increase the weightage λ for the spectral loss. As the margin for improvement is least for the LoConv model, spectra in Figure 9a have the least changes. Specifically, notice in panel (a) that the gap between the blue line, representing LoConv- λ_0 predicted

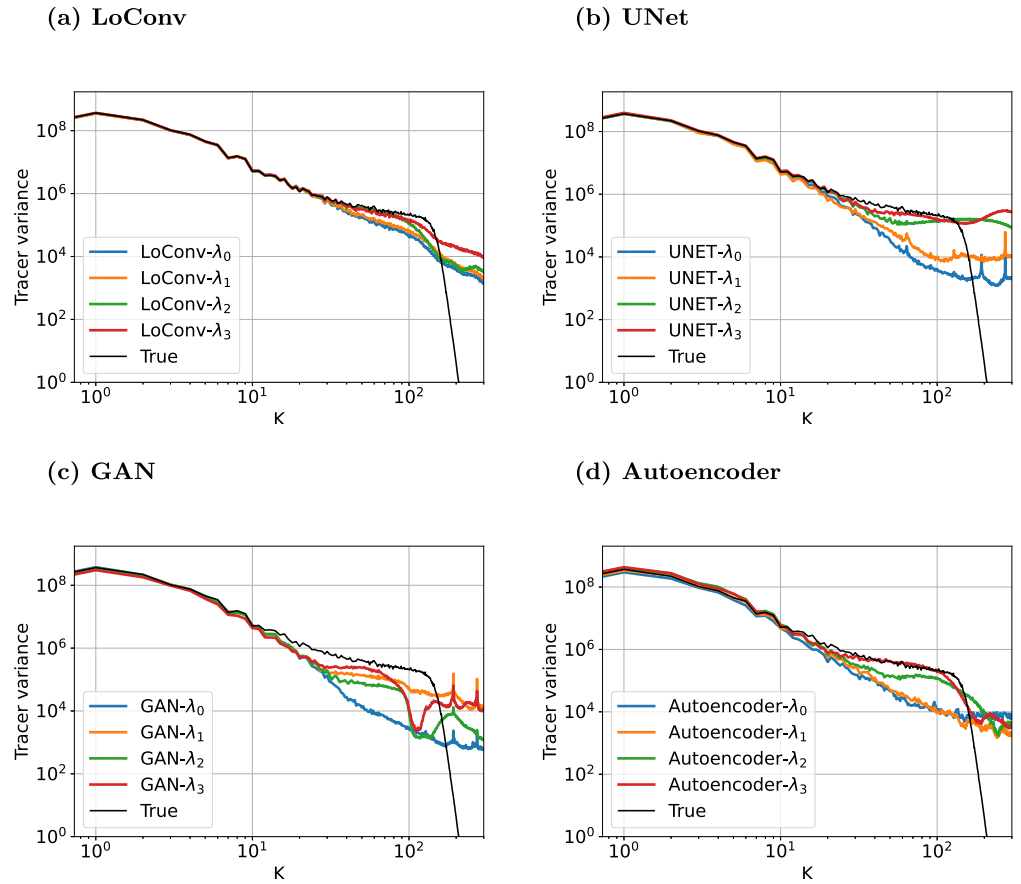


Figure 9. Comparison of tracer variance spectrum predictions with varying λ for $Ro = 0.7$ and $E_C/E_T = 0.15$.

spectrum, and the black line, showing the true variance, is minimal in the inertial range. Similarly, the UNet spectra shown in Figure 9b improves monotonically with increasing spectral weight, covering a larger gap between the UNet- λ_0 predicted spectrum in blue and the true spectrum in black. In Figure 9c, the GAN spectrum improves slightly in the inertial range but diverges slightly at larger scales or small wavenumbers. The spectrum of GAN also starts to fluctuate near the dissipation range after the introduction of spectral loss. Autoencoder spectra also improve monotonically, but does not match perfectly with the true spectrum at larger scales.

Let us return to the effect of adding spectral loss on the physical structure of predicted tracers. The LoConv model shows improvement in producing small-scale dispersive structure, as seen by comparing Figures 7b and 8b. However in the case of UNet, GAN, and Autoencoder, adding spectral loss produces the artificial sharp dot-like structure where the sign of the tracer field value fluctuates concentrically around the dot region. Similarly, there is not much improvement in the production of small-scale dispersive structures by these models. The dot-like structure is very small but can change the Fourier coefficients for high modes. Adding such a structure drastically improves the spectrum of the prediction while not having much qualitative change in the regions away from the dot-like structure. Therefore, the UNet, GAN, and Autoencoder models achieve a better match in spectra at the cost of losing physical structure match due to the production of artificial dots.

Recall that we are based in a doubly periodic domain for the flow and tracer fields and one could enforce periodicity constraints during the training phase. The different means for this include using circular padding, training in the Fourier domain with convolution layers, and using complex valued layers, as in Bassey et al. (2021), to ensure periodicity in predictions. On experimenting with these and related techniques, some methods led to inferior predictions in the physical domain while other methods ended up being much more intense computationally. Compared to these strategies, not enforcing any periodicity constraint during the training phase, expecting the neural networks to learn the periodic features, worked best in our cases. The departure from periodicity was quite small in our predictions and this is quantified in Text S1 and Figure S7 in Supporting

Information S1. All the results presented in this work were therefore generated without enforcing any periodicity constraints in training, since this strategy already gave us good predictions.

5.4. Tracer Flux Computed From the Predictions

The tracer variance is transferred from scale to scale at a rate dictated by the tracer flux. Here we will derive an expression for tracer flux in physical space and compute it using the tracer predictions. Since tracer flux is a secondary derived quantity, calculated once the tracer field is known, comparing flux in physical space from different models serves as a more stringent test for the capability of the models.

We define a filtering operator giving $\tilde{\phi}$ on operating on any field ϕ such that

$$\tilde{\phi} = F^{-1} \hat{\phi}_{k < \tilde{k}} \quad (8)$$

Above $\hat{\phi}$ represents the Fourier transform of ϕ , F^{-1} represents inverse Fourier transform operator and the subscript denotes that all Fourier coefficients except for those with wavenumbers $k < \tilde{k}$ are set to zero. The filtering operator therefore retains the large-scale part of the tracer field.

Applying the filter operator to the tracer Equation 2 after ignoring the forcing and dissipation terms leads to

$$\frac{\partial \tilde{\theta}}{\partial t} = -(\tilde{\mathbf{v}} \cdot \nabla \tilde{\theta}) \quad (9)$$

Adding $\tilde{\mathbf{v}} \cdot \nabla \tilde{\theta}$ on both sides and using the incompressibility condition on the filtered velocity, $\nabla \cdot \tilde{\mathbf{v}} = 0$, gives

$$\frac{\partial \tilde{\theta}}{\partial t} + \tilde{\mathbf{v}} \cdot \nabla \tilde{\theta} = \nabla \cdot (\tilde{\mathbf{v}} \tilde{\theta} - (\tilde{\mathbf{v}} \tilde{\theta})) \quad (10)$$

Multiplying throughout with $\tilde{\theta}$ and manipulation gives us

$$\frac{\partial}{\partial t} \left(\frac{\tilde{\theta}^2}{2} \right) + \tilde{\mathbf{v}} \cdot \nabla \left(\frac{\tilde{\theta}^2}{2} \right) = \tilde{\theta} \nabla \cdot (\tilde{\mathbf{v}} \tilde{\theta} - (\tilde{\mathbf{v}} \tilde{\theta})) = -\Pi \quad (11)$$

The quantity $\tilde{\theta}^2$ is the total tracer variance contained in large scales or $k < \tilde{k}$ and it is transferred to smaller scales due to the right hand side term above, which we will identify as tracer flux, Π . This is seen more explicitly by integrating Equation 11 over the bi-periodic domain (D) to get

$$\frac{d}{dt} \int_D \left(\frac{\tilde{\theta}^2}{2} \right) dx = - \int_D \Pi dx \quad (12)$$

Notice above that domain-integrated $\tilde{\theta}^2$ changes due to the domain-integrated flux. The negative sign appears due to tracer variance being lost from large to small scales as a result of the turbulent cascade.

We will now look at the flux derived from each of the models. Specifically, we will present results for the case $\tilde{k} = 64$, this being an arbitrary wavenumber in the inertial range, as seen in the spectra shown in Figures 5–9. Other wavenumbers led to similar conclusions as those given below and therefore will not be presented here.

Figure 10 shows the true flux derived from the true tracer field corresponding to $Ro = 0.3$ and $E_C/E_T = 0.15$ in panel (a) and the flux derived from predictions of all models trained with no spectral loss in panels (b) to (e). We note that all model predictions other than the LoConv model are significantly inaccurate and do not get the major structures right. The LoConv- λ_0 model prediction in Figure 10b gets all significant structures in the flux field but looks lighter, that is, has lower magnitude than the true flux. This can be seen, for example, by comparing the region in the black box in panel (a) of Figure 10 with the same region in panel (b). The structure inside this box is significantly faded in other predictions, while the LoConv prediction gets the structure with some fading. This is

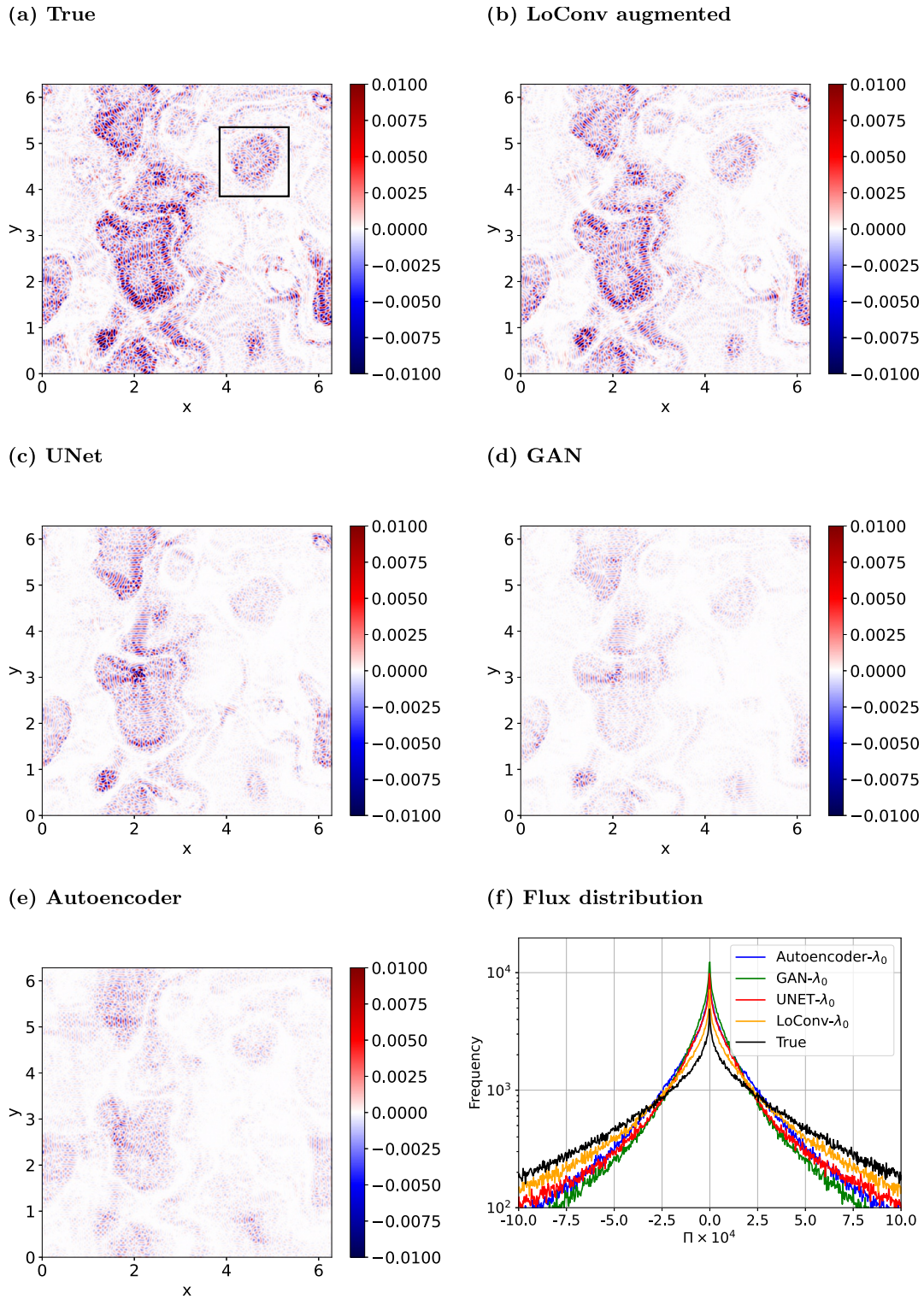


Figure 10. Comparison of predictions of physical structure of tracer variance flux and distributions from models with $\lambda = \lambda_0$ for $Ro = 0.3$ and $E_C/E_T = 0.15$. The black box highlights a specific region for comparison.

also evident from the distribution of flux shown in Figure 10f. In the distribution we can see that the frequency of larger values of flux (away from center or at the tails) obtained by predictions is underestimated compared to the true flux. On the other hand, smaller values of flux have more frequency in the predictions than they should be according to true flux. Among the λ_0 models, the predicted distributions of the LoConv- λ_0 model match the actual distribution the best, followed by UNet- λ_0 , then Autoencoder- λ_0 , and finally GAN- λ_0 . It is also noteworthy that the Autoencoder's flux prediction is comparable to that of GAN, although GAN had a lower pointwise error for the tracer field prediction, seen earlier.

Figure 11 contains predictions for $Ro = 0.3$ and $E_C/E_T = 0.15$ from all models that have been trained with maximum weightage of spectral loss, that is, $\lambda = \lambda_3$. We see that the LoConv- λ_3 model prediction in Figure 11b is much better now. Notably, the prediction is similar in intensity and does not look lighter than the true flux, which was the case when no spectral loss was used in Figure 10b. However, the GAN- λ_3 and UNet- λ_3 predictions in the second row are much worse when using spectral loss with $\lambda = \lambda_3$. There is the presence of concentric flux regions around a central dot, these being highlighted by yellow boxes in panels (c), (d), and (e). These artificial features are due to the presence of the large dot-like structures in the tracer prediction itself, seen in panels (c), (d), and (e) of Figures 6 and 8. This causes a lot of flux to be concentrated in the region around the dot, in turn producing a lighter prediction in the field everywhere else. The distribution curves of all predicted flux seem to converge to the true distribution in Figure 11f, with LoConv- λ_3 being the closest in orange, followed by Autoencoder- λ_3 in blue, then the UNet- λ_3 in red, and, furthest, the GAN- λ_3 in green.

The flux for $Ro = 0.7$ and $E_C/E_T = 0.15$ from different models is shown in Figure 12 without spectral loss, that is $\lambda = \lambda_0$, and these are qualitatively similar to the lower Rossby number case discussed earlier in Figure 10. Specifically, except LoConv, the predictions are much lower in magnitude. LoConv gets the key structures right albeit being a bit lower in magnitude. Correspondingly, the flux distributions in panel (f) shows that LoConv aligns relatively better with the true flux distribution, although the LoConv predicted frequency overshoots the low flux region frequency from the true distribution. Other models' distributions are much further from the true distribution. Interestingly, Autoencoder- λ_0 and UNet- λ_0 have similar distributions, despite UNet- λ_0 having far less absolute error in the prediction of the tracer field, as discussed earlier. This reaffirms that having lower absolute pointwise error does not necessarily correspond to better overall prediction.

Figure 13 shows the same regime, $Ro = 0.7$ and $E_C/E_T = 0.15$, but for models trained with $\lambda = \lambda_3$. From the top row and panel (f) we see that the LoConv prediction matches the tracer flux physical structure and distribution very well. UNet, GAN, and Autoencoder however produce artificial concentrated regions of high flux, these regions being highlighted in yellow boxes in the figure. Additionally, their distributions do not align well with the true flux distribution, although all the three models more or less are similar in terms of their distributions.

To specifically highlight the effect of weightage on distributions, Figure 14 shows the flux distributions for different λ for $Ro = 0.7$ and $E_C/E_T = 0.15$. We see that the distributions of predicted flux converge closer to the distribution of the actual tracer flux for all model architectures as we increase the weightage of spectral loss. However, only LoConv gets the low flux region's frequency close to the truth. Other models' predictions are seen to overshoot the truth in the low flux region. As for the tracer structures and spectra, we omit more figures for other regimes here, but have incorporated the flux derived from predictions for $Ro = 0.3$ and $E_C/E_T = 0.01$, from all the models along with a comparison of flux distributions, in Figures S8 and S9 in Supporting Information S1. The Supporting Information S1 also contains similar figures for $Ro = 1$ and $E_C/E_T = 0.15$ in Figures S10 and S11 in Supporting Information S1, corresponding to models trained with no spectral loss and maximum spectral loss, respectively.

Similar to the tracer field, we computed pointwise error of flux from different models and found that LoConv had the lowest error. In addition, we examined the Pearson correlation coefficient of the predicted and true flux, this being a more useful variable that checks the relative alignment of structures in the data. The mean of Pearson correlation across samples from different Rossby numbers and energy ratios can be used to compare the flux prediction across models. The mean Pearson correlation values of the models are given in Figure S12 in Supporting Information S1. The LoConv models in general had mean Pearson correlation over 0.815, while other models had Pearson correlation under 0.64. The LoConv models producing best small-scale dispersive structures are LoConv- λ_2 and LoConv- λ_3 . LoConv- λ_2 has mean correlation 0.816, and LoConv- λ_3 has correlation 0.824, a slight improvement.

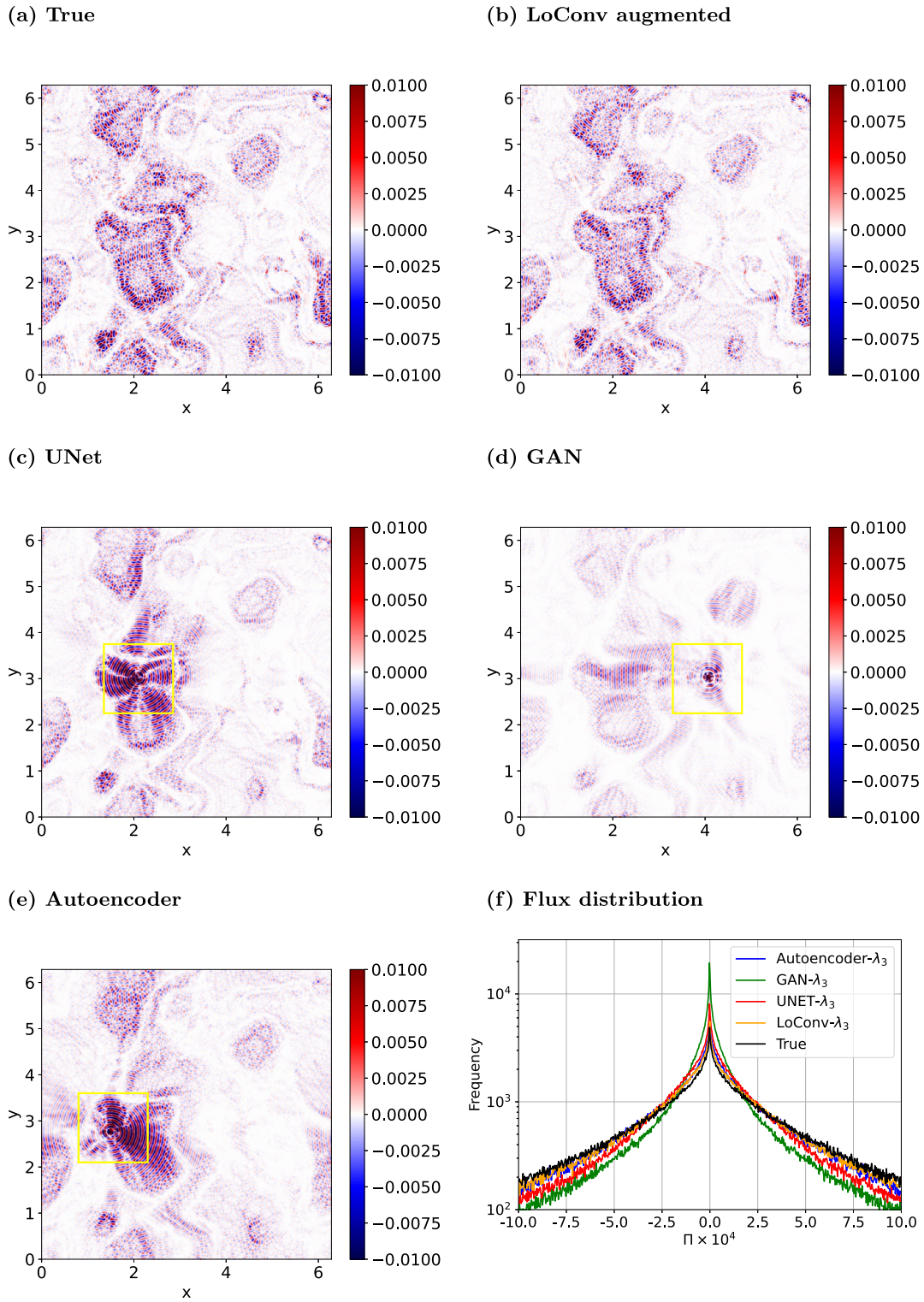


Figure 11. Comparison of predictions of physical structure of tracer variance flux and distributions from models with $\lambda = \lambda_3$ for $Ro = 0.3$ and $E_C/E_T = 0.15$. Yellow boxes contain a region with anomalous physical features.

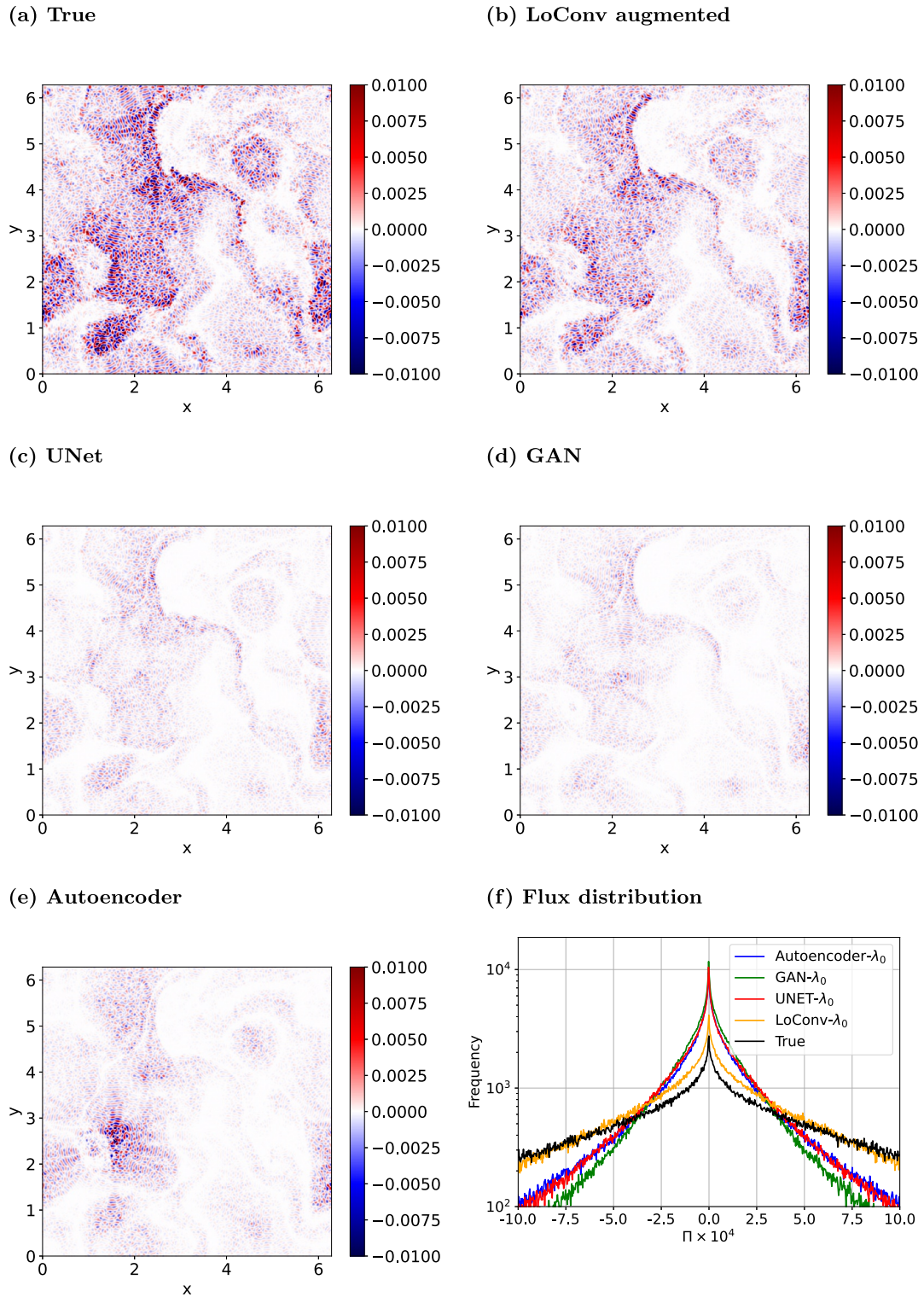


Figure 12. Comparison of predictions of physical structure of tracer variance flux and distributions from models with $\lambda = \lambda_0$ for $Ro = 0.7$ and $E_C/E_T = 0.15$.

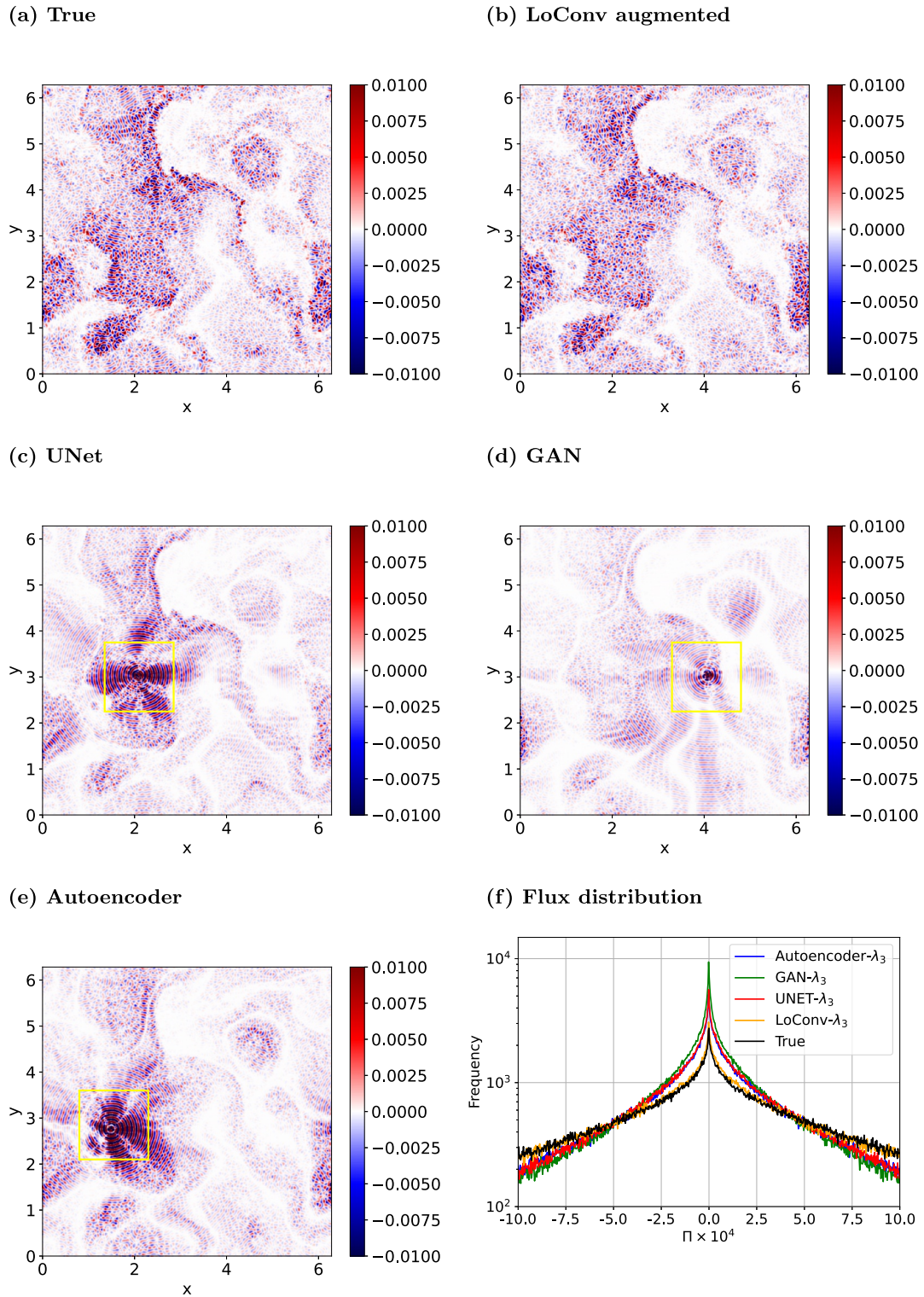


Figure 13. Comparison of predictions of physical structure of tracer variance flux and distributions from models with $\lambda = \lambda_3$ for $Ro = 0.7$ and $E_C/E_T = 0.15$. Yellow boxes contain a region with anomalous physical features.

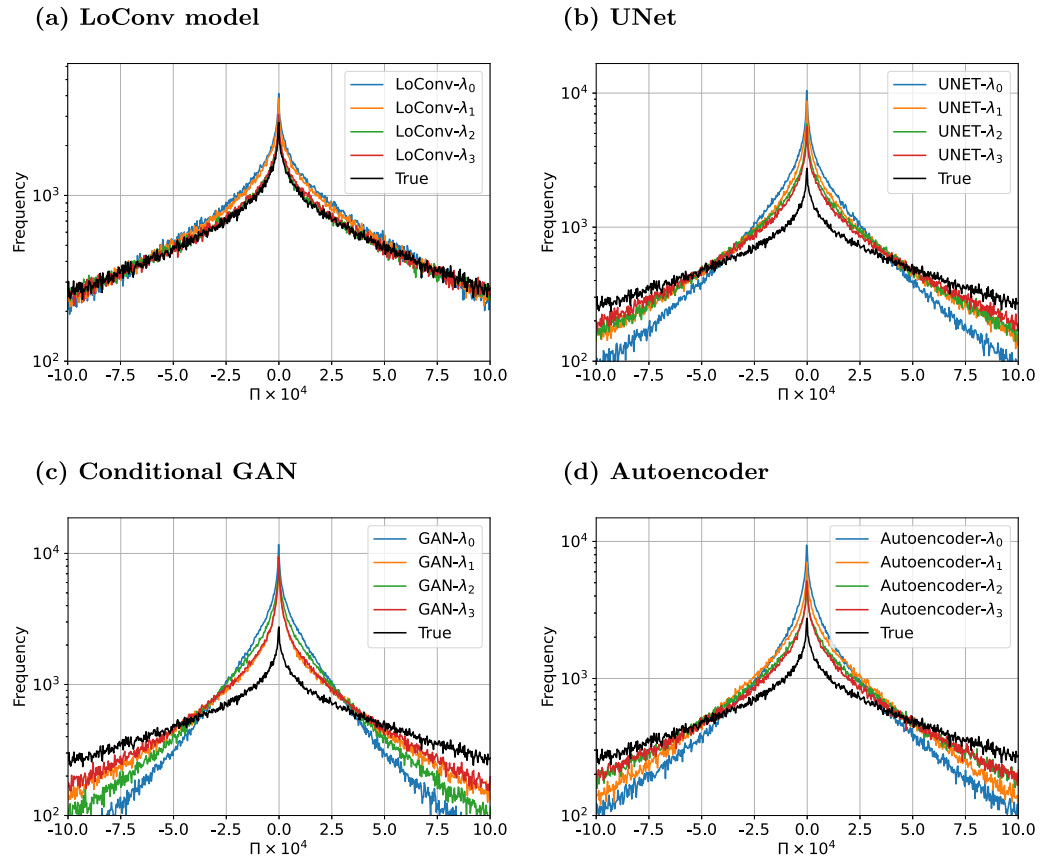


Figure 14. Distribution of tracer variance flux with varying weightage of spectral loss for $Ro = 0.7$ and $E_C/E_T = 0.15$.

To summarize the comparisons between models, we see that prediction of small-scale tracer structures is very important as it helps in accurately setting the flux. Considering the different metrics to evaluate our models, LoConv models unanimously come on top. Among the LoConv models, LoConv- λ_2 and LoConv- λ_3 have the best spectrum prediction, with almost overlapping spectra, as seen in Figure 9. The LoConv model with $\lambda = \lambda_3$ has the most visually similar-looking tracer structures compared to the true tracer. It also has slightly better flux prediction. Given these, we declare the LoConv- λ_3 model as the best from the ones we tested and proceed to a specific application using this model in the next section.

5.5. An Application of Neural Network Prediction

Given the tracer predictions obtained from the neural networks, we now turn to a specific test following a result from Sirohi and Thomas (2024). Using the tracer flux physical space distribution, Sirohi and Thomas were able to identify physical space regions that accommodated the dominant share of the tracer flux. This is useful information, especially for in situ measurement programs for instance, where there is advantage in identifying the specific spatial locations where the tracer gets transferred to smaller scales. This information can then be used along with other flow details found from measurements, such as fronts, filaments, etc in the region of exploration and thereby helps to connect and associate tracer cascades with specific flow features. Here, we will check how well such regions can be predicted by the best neural network model seen so far: LoConv- λ_3 .

Following the same procedure as in Sirohi and Thomas (2024), we sorted the flux Π against normalized vorticity ζ/ζ_{\max} , ζ_{\max} being the maximum vorticity value at a particular instant. This gives us $\Pi(\zeta/\zeta_{\max})$, that is, a one-dimensional array of flux associated with a one-dimensional array of normalized vorticity. We then performed a cumulative sum of the sorted flux as

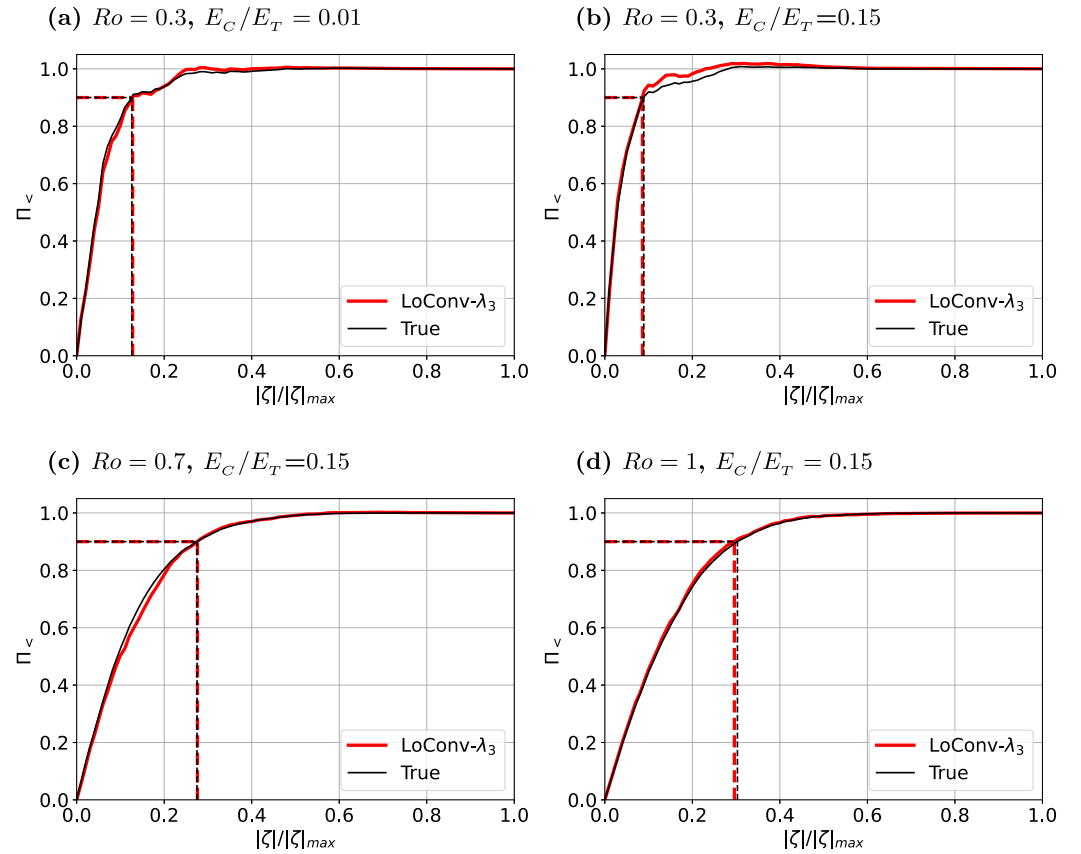


Figure 15. Cumulative flux, $\Pi_{<}$ from Equation 13, against normalized vorticity is plotted for four different regimes. In each case the neural net prediction is shown by the red curve while the true tracer based curve is in black color. Dashed horizontal lines corresponding to each curve shows the 90 percent flux, this being dropped onto the x-axis by dashed vertical lines.

$$\Pi_{<}(\gamma) = \frac{\iint_D \Pi \cdot \mathcal{H}\left(\frac{|\zeta|}{|\zeta|_{\max}} < \gamma\right) dx}{\int_D \Pi dx} \quad (13)$$

Above \mathcal{H} is the heavy-side function, which takes the value 1 if the condition inside it is true and 0 otherwise. \mathcal{H} is used to separate out regions in the vorticity field below a given value. The function $\Pi_{<}$ gives us the fraction of flux contained in regions where the vorticity magnitude is below a certain threshold, γ .

We obtained the cumulative flux $\Pi_{<}$ from the true tracer and the LoConv- λ_3 prediction and these are compared in Figure 15 for four different Rossby number-energy ratio combinations. The plotted curves for each Rossby number and energy ratio are averages across 10 instantaneous fields. The black curves are obtained using flux from the true tracer field and the red curves are from flux computed using the LoConv- λ_3 predicted tracer field. In each panel we have added dashed lines in red and black, representing the value of vorticity at which 0.9 fraction of total flux is obtained. We will call this point the 90-percent point. To obtain the 90-percent point, we arranged the values of $\Pi_{<}$ with increasing values of γ , and then chose the value of γ for which $\Pi_{<}(\gamma)$ crosses 0.9 for the first time, and declared that as the 90-percent point. We note that the cumulative flux plots from the model and truth match very well, and the 90% marker in the black dashed lines almost overlaps the red dashed lines in all cases.

The plot of cumulative flux for Rossby number $Ro = 0.3$ and energy ratio $E_C/E_T = 0.01$ are presented in panel (a) of Figure 15. In this case the 90% of the true flux is contained in a region where the vorticity is below 0.126 of its maximum magnitude, that is, the true 90-percentile point is 0.126, whereas the predicted 90-percentile point is 0.128. Increasing the energy ratio to $E_C/E_T = 0.15$ for $Ro = 0.3$ changes the true 90%-percentile value to 0.089, and the prediction of the percentile value in this case is 0.086, as shown in panel (b) of Figure 15. We note that

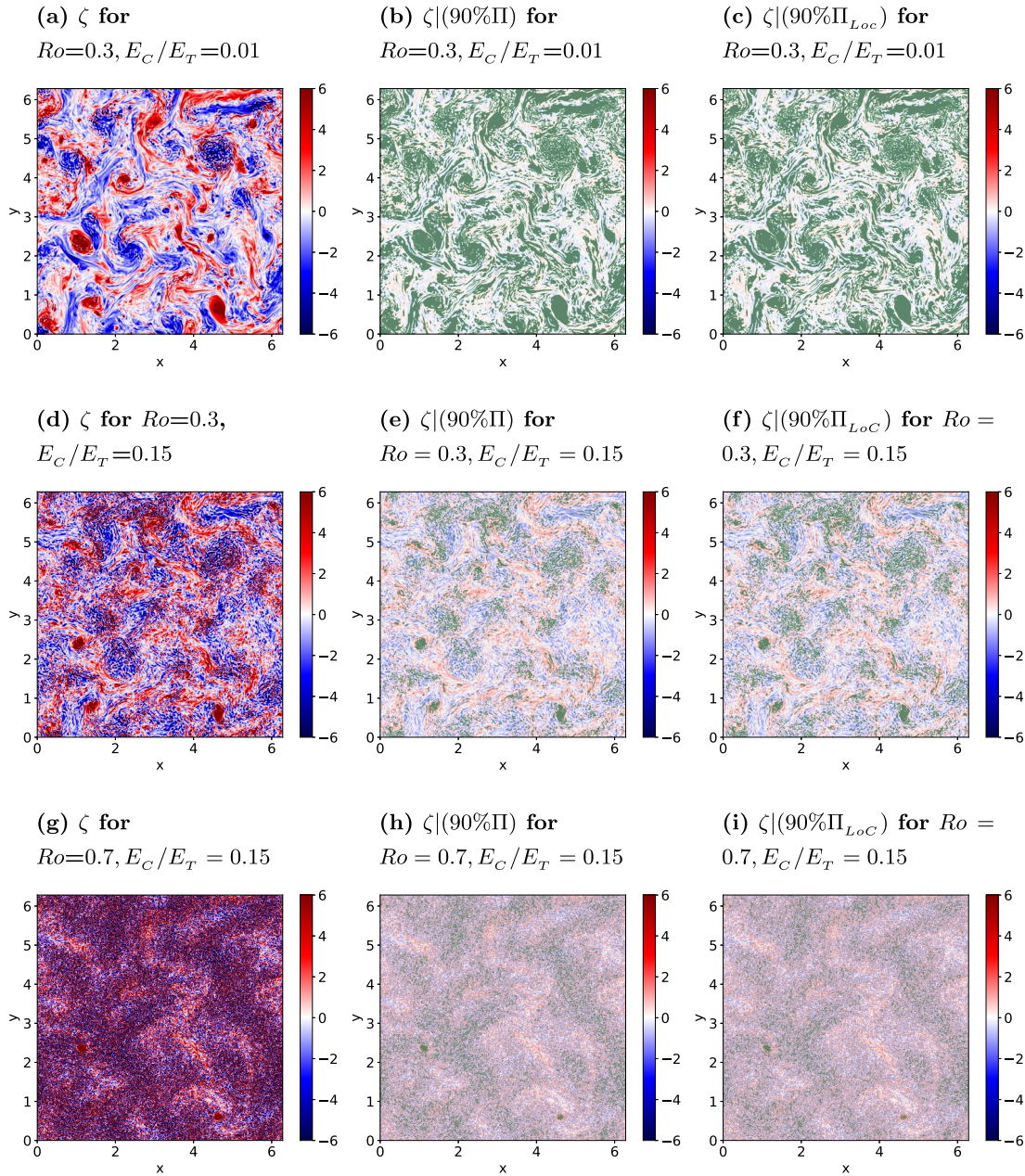


Figure 16. Left column shows the full vorticity field. From this, vorticity regions that contain 90% of tracer flux computed using the true tracer in shown middle column, these regions being dimmed in color while green colored region shows the excluded regions that are not part of the 90% flux regions. Right column shows the same quantity as the middle column, but computed using tracer predicted by the LoConv model with $\lambda = \lambda_3$.

increasing the energy ratio actually lowers the 90-percentile point. For $Ro = 0.7$ and $E_C/E_T = 0.15$, the true 90%-percentile value is 0.275, and the prediction is 0.276, shown in Figure 15c. For $Ro = 1.0$ and $E_C/E_T = 0.15$, the true 90%-percentile value is 0.303 and the prediction is 0.296, as shown in Figure 15d. In general we find that the LoConv- λ_3 predictions of the 90-percentile values match the true value with great precision.

Once we have the vorticity values that accommodate 90% flux, we can locate those vorticity regions in physical space and this is done in Figure 16. The left column shows the vorticity field for three regimes, these figures being the same ones shown in the left column of Figure 1. The middle column shows the vorticity regions from the true fields that contain 90% flux, the red and blue colors being slightly dimmed here. In green color we have marked regions that are excluded and do not contribute towards the 90% flux regions. In other words, the red and blue

regions excluding the green regions constitute physical space regions that accommodate 90% flux. Finally, the right column shows the same field as the middle columns, based on the neural network prediction. In the top most row we see that the excluded regions are primarily large coherent vortices. This means that most of the tracer flux is located outside coherent vortices, in strain dominant regions, these being active tracer stirring sites. As we go down, the size of coherent vortices decreases and the green regions get more scattered across the domain. These results are discussed in detail in Sirohi and Thomas (2024), although the highlight here is being able to predict dominant tracer flux regions from the neural network output tracer field. Simply put, getting the second column of Figure 16 requires quite a bit of work, such as integrating the tracer advection equation along with the flow equations followed by computing the flux from the data and sorting it further to get 90% regions. However, just by knowing the instantaneous flow field, we are able to generate the third column of Figure 16 using the neural network prediction of the tracer field, without integrating the tracer equation.

6. Summary

In this work we investigated the possibility of predicting passive tracer dispersion by submesoscale flows using neural networks. The flow was modeled using a two-vertical-mode model and we used the divergence-free barotropic flow to advect a passive tracer across different Rossby numbers and baroclinic-barotropic energy ratios. Much of the physical understanding of the flow and tracer dispersion using this model is discussed in Thomas and Vishnu (2022) and Sirohi and Thomas (2024), but by integrating the flow and tracer equations simultaneously. Our goal in this work was to develop neural network models to predict the tracer field and its related quantities, such as spectra, flux, relevant physical space regions, etc from the knowledge of the flow alone, without separately time integrating a tracer equation.

We used four types of neural network architectures: Autoencoder, UNet, GAN, and LoConv. These models take in the vorticity field and predict the tracer field. Each of these were trained with four different training loss functions. The difference came from the weightage given to spectral loss relative to pointwise absolute loss and was described by a parameter λ . The purpose of the spectral loss was to rectify the predicted tracer spectral variance in the inertial range. Each combination of four architectures and four values of parameter $\lambda \in \{\lambda_0 < \lambda_1 < \lambda_2 < \lambda_3\}$ gave us 16 models in total. Note that all architectures other than the LoConv model are popular architectures based on convolution layers. We designed the LoConv models using custom *Local Convolution* layers that act like convolution layers, but their filter weights change as the convolution window moves in space. This was done with the intention of capturing spatially inhomogeneous dynamics. We then compared the models' predictions for various Rossby numbers and energy ratios for varying weightage of spectral loss.

Based on physical structures and point-wise absolute errors, the model architectures can be ranked from worst to best as Autoencoder, GAN, UNet, and then LoConv. The Autoencoder did not capture the large-scale tracer structures accurately; the model was seen to create patches of low intensity regions within coherent tracer structures. The prediction quality of Autoencoder was seen to improve as we increased the weightage of spectral error λ . The GAN and UNet models both captured the large-scale tracer structures very accurately, but they failed to capture small-scale features. This was also seen in the spectral tracer variance associated with their predictions, the variance spectra of the GAN and UNet predictions were seen to fall off at small-scales, often faster than the Autoencoder. As we increased λ , the inertial range spectra of the UNet and GAN models improved significantly, but at the expense of loss of coherence in physical structures. The GAN predictions got smoother, losing small-scale features, while there was only a slight improvement in the production of small-scale features in the UNet predictions. Autoencoder showed some improvement in small-scale structure prediction, although these features were not realistic as seen in the true tracer field. In addition, sharp dot-like artificial structures appeared in the predictions of GAN, UNet, and Autoencoder, along with concentric, fluctuating signs of the tracer field around it. Finally, the LoConv model precisely predicted the large-scale tracer structures and had the least deviation of the spectrum away from the true spectrum. This model had the best small-scale and large-scale predictions among all models. With increasing weightage for spectral loss, the LoConv models got rid of the small discrepancy in spectrum and also had increased production of small-scale features, reflecting a realistic tracer field.

The importance of correctly capturing small-scale tracer features was clear when looking at the tracer flux predictions, this being a derived quantity from tracer predictions. Flux predictions of all models except LoConv

were significantly inaccurate. Without spectral weight, all model-predicted fluxes were lighter, that is, of lower magnitude compared to the true flux. As we increased the spectral loss weightage, the LoConv model's predictions got better. They generated the large and small-scale structures with the correct magnitude. In the case of GAN, UNet, and Autoencoder, the magnitude of the flux improved, but along with inaccurate structures in physical space. There were concentric circular patches of flux around dot-like structures that were produced in the predicted tracer field. Despite having inaccuracies in large-scale structures, the Autoencoder had better small-scale tracer structures compared to GAN and also sometimes compared to UNet, specifically when the models were trained without any spectral loss. This was also seen to reflect in the flux predictions. Overall, the LoConv model trained with maximum spectral loss weightage λ_3 had the best predictions of the flux field. We then used this model to predict the physical space regions that contained 90% of the tracer flux and these predictions agreed very well with those obtained from the actual tracer field, reinstating the advantages of such neural network models that can directly generate tracer fields once the flow is known.

The ability of neural networks to predict tracer dispersion means that we have a direct functional way of obtaining the tracer field, avoiding numerical integration of the tracer advection equation. As described quantitatively in Section 4, this can be a great advantage when tracer predictions are needed for a large set of regimes. In such cases the computational time needed for direct numerical integration of the tracer equation far exceeds the time needed for neural networks. Also recall that in the introduction we saw that the number of relevant tracers in an ocean model can exceed a dozen, and it will be a great achievement if all or most of those tracers can be directly predicted from the flow using pre-trained neural networks. The LoConv model developed in this study could be an attractive choice in this direction. LoConv captures local patterns or relations that are highly heterogeneous in space, this being common in oceanic flows in the presence of bottom topographies and continental boundaries (Brady et al., 2021; Kunze & Smith, 2004; Moum et al., 2002; Wunsch, 1970).

Despite the prospects for future discussed above, we remind the reader that our present work used an idealized set up, while in the real ocean there are more challenges. To examine the possibility of predicting tracer fields, we chose a two-dimensional flow set up based on the two-vertical-mode model. Predicting three-dimensional tracer fields from three-dimensional flow fields is possible by continuing a similar set-up as ours by simply replacing two-dimensional convolution with three-dimensional convolution layers, albeit needing higher computational facility to train the neural networks, as the model size would increase by a factor of the convolution window size in depth direction, and the ram usage for one forward passage through the neural network and the inference time would both increase linearly with depth grid size. Additionally, our work was restricted to passive tracers, while there are many important tracers in the oceans that are active, that is tracers that impact the evolution of the flow field, such as salinity and heat (Corrège, 2006; Jensen, 2001). There are also reactive tracers that interact among themselves and thereby affect the evolution of the tracers. Planktons that form important ecosystems in the oceans are an example (Mahadevan, 2016).

One could imagine using online learning with the PDE evolution, for instance, as in Frezat et al. (2022) or Kochkov et al. (2024), to help in the prediction of active and reactive tracer structures. Additionally, the methods applied in this work can be extended to predict multiple tracers by increasing the number of output channels. If these are realized in the near-future, and we get to predict over a dozen tracer fields, including passive, active, and reactive ones, in three-dimensional ocean model set-ups, that would indeed be a great achievement in the field of data-driven ocean modeling. We hope that future follow-up of this study can move in that direction.

Appendix A: Flow of Data Through the Neural Networks

In Figure 3, above each icon representing a layer, we have indicated the number of output channels, feature window size, and stride length as a triad of numbers. The neural network models take input of vorticity in the form of 768×768 arrays and produce output tracer fields as arrays of the same shape. The input is always converted to a $768 \times 768 \times 1$ array to add a channel dimension. After passing through a convolution layer or a down-sampling layer, the spatial dimension reduces by a factor of strides. Passing through a convolution transpose or an up-sampling layer, the opposite happens, and the spatial dimension increases by a factor of strides. The channel dimensions become as specified in the layer.

In figure 3a, a $768 \times 768 \times 1$ input array passes the first down-sampling layer (red icon) of the Autoencoder. As specified above this icon, the number of output channels, feature window length, and strides for this layer are 64,

16, and 8, respectively. The output of this layer has a shape in the form of output length \times output breadth \times output channels. The number of output channels is 64, and length or breadth in the output is obtained by reducing the input length by a factor of strides, that is, $768/8 = 96$. So the shape of the output of this layer is $96 \times 96 \times 64$. Passing the second layer, it becomes $12 \times 12 \times 128$. Then passing the first green icon up-sampling layer, its resolution increases back, and its shape becomes $96 \times 96 \times 128$. It then passes a convolution transpose layer with one output channel, indicated by a gray icon. This layer glues together the features learned in the previous layer and produces an output of shape $768 \times 768 \times 1$. This represents the predicted tracer.

Similarly, in Figure 3(b.1), the diagram represents the UNet as well as the generator of the conditional GAN. Here again the $768 \times 768 \times 1$ shaped vorticity input after passing through the first down-sampling layer indicated by the top most red icon on the left of panel (b.1), takes shape $192 \times 192 \times 256$. Then passing through another down-sampling layer (second from top red icon in panel (b.1)), the shape is $64 \times 64 \times 256$. Consecutively passing through the down sampling layer, going down the red icon ladder on the left side of panel (b.1), the extracted output reaches the second last down-sampling layer, where the output, let's call it D_{-2} , has shape $8 \times 8 \times 512$. Then after passing the bottom most down-sampling block shown by red icon in the middle of the red and green ladders in panel (b.1) the output takes shape of $2 \times 2 \times 1024$. After this it passes through an up-sampling block (bottom green icon on left of panel (b.1)), and the shape is $8 \times 8 \times 1024$. This is then concatenated with D_{-2} , the output of the penultimate down-sampling layer, along the channel dimension. We add both their number of channels to get the number of channels in the concatenated output. This comes out to be $512 + 1024 = 1536$, resulting in shape $8 \times 8 \times 1536$. Then after repeated up-sampling and concatenation, a shape $192 \times 192 \times 512$ array is obtained. This is passed through a final convolution transpose layer indicated by gray icon on the top right hand side of panel (b.1), and the output of shape $768 \times 768 \times 1$ is obtained. We skip describing the data flow in the discriminator, as the adversarial movement of outputs has already been explained in Section 3.6. Moreover, it is a simple convolutional down-sampling neural network, and is only used in training.

As mentioned in the neural network section, LoConv model has a UNet type base along with two LoConv layers concatenated in between. A similar process of down-sampling, then up-sampling, and concatenation happens in the LoConv part indicated by the red and green icons' ladder in the lower part of panel (c) of Figure 3. Additionally, the initial vorticity input is passed through two LoConv layers indicated by blue icons at the top of panel (c). One of them has a stride of length one and eight output channels. The output \mathcal{L}_1 of this layer has shape $768 \times 768 \times 8$. The other layer has strides of length four and 32 output channels. The output \mathcal{L}_2 of this layer is of shape $192 \times 192 \times 32$. The array \mathcal{L}_2 is then concatenated with the output of the final up-sampling layer (top green icon on the right). The number of channels here is obtained by adding the number of channels coming from each of the concatenating arrays, which is $64 + 64 + 32 = 160$. The concatenated output is of shape $192 \times 192 \times 160$. This is passed through a convolution transpose layer (second from top gray icon in panel (c)) to get an output of shape $768 \times 768 \times 8$. Again this output is concatenated with \mathcal{L}_1 with the concatenated array having shape $768 \times 768 \times (16 = 8 + 8)$ and passed through a convolution transpose layer (top green icon in panel (c)) with one output channel and strides of length one to get an output of shape $768 \times 768 \times 1$.

Data Availability Statement

The data and scripts used for this manuscript is available at Bijay and Thomas (2025).

Acknowledgments

MKB thanks the PhD program in mathematics run jointly by ICTS-TIFR and TIFR-CAM. JT acknowledges the Deep Ocean Mission scheme of the Ministry of Earth Sciences for providing financial support for this work through the project F.No.MoES/PAMC/DOM/18/2022 (E-12926). The authors also acknowledge discussions that benefited this work from two meetings organized at ICTS-TIFR: *Mathematical modeling of Climate, Ocean, and Atmosphere processes* (ICTS/COAPS2023/04) and *Theoretical and Practical Perspectives in Geophysical Fluid Dynamics* (ICTS/TAPGFD2024/05).

References

- Abernathy, R. P., & Marshall, J. (2013). Global surface eddy diffusivities derived from satellite altimetry. *Journal of Geophysical Research: Oceans*, 118(2), 901–916. <https://doi.org/10.1002/jgrc.20066>
- Alford, M., MacKinnon, J., Simmons, H., & Nash, J. (2016). Near-inertial internal gravity waves in the ocean. *Annual Review of Marine Science*, 8(1), 95–123. <https://doi.org/10.1146/annurev-marine-010814-015746>
- Bassey, J., Qian, L., & Li, X. (2021). A survey of complex-valued neural networks.
- Bianchi, D., Dunne, J. P., Sarmiento, J. L., & Galbraith, E. D. (2012). Data-based estimates of suboxia, denitrification, and N_2O production in the ocean and their sensitivities to dissolved O_2 . *Global Biogeochemical Cycles*, 26(2). <https://doi.org/10.1029/2011gb004209>
- Bijay, M. K., & Thomas, J. (2025). 2025JH000655 [Dataset]. *Zenodo*. <https://doi.org/10.5281/zenodo.15305424>
- Bolton, T., & Zanna, L. (2019). Applications of deep learning to ocean data inference and subgrid parameterization. *Journal of Advances in Modeling Earth Systems*, 11(1), 376–399. <https://doi.org/10.1029/2018ms001472>
- Booth, J., & Kamenkovich, I. (2008). Isolating the role of mesoscale eddies in mixing of a passive tracer in an eddy resolving model. *Journal of Geophysical Research*, 113(C5). <https://doi.org/10.1029/2007jc004510>
- Brady, R. X., Maltrud, M. E., Wolfram, P. J., Drake, H. F., & Lovenduski, N. S. (2021). The influence of ocean topography on the upwelling of carbon in the southern ocean. *Geophysical Research Letters*, 48(19), e2021GL095088. <https://doi.org/10.1029/2021gl095088>

- Busecke, J. J. M., & Abernathey, R. P. (2019). Ocean mesoscale mixing linked to climate variability. *Science Advances*, 5(1), eaav5014. <https://doi.org/10.1126/sciadv.aav5014>
- Capet, X., McWilliams, J. C., Molemaker, M. J., & Shchepetkin, A. F. (2008). Mesoscale to submesoscale transition in the California current system: Energy balance and flux. *Journal of Physical Oceanography*, 38(10), 2256–2269. <https://doi.org/10.1175/2008jpo3810.1>
- Chouksey, A., Griesel, A., Chouksey, M., & Eden, C. (2022). Changes in global ocean circulation due to isopycnal diffusion. *Journal of Physical Oceanography*, 52(9), 2219–2235. <https://doi.org/10.1175/jpo-d-21-0205.1>
- Corrège, T. (2006). Sea surface temperature and salinity reconstruction from coral geochemical tracers. *Palaeogeography, Palaeoclimatology, Palaeoecology*, 232(2–4), 408–428. <https://doi.org/10.1016/j.palaeo.2005.10.014>
- Ducournau, A., & Fablet, R. (2016). Deep learning for ocean remote sensing: An application of convolutional neural networks for super-resolution on satellite-derived SST data. In *2016 9th IAPR Workshop on Pattern Recognition in Remote Sensing (PRRS)* (pp. 1–6).
- Ferrari, R., & Wunsch, C. (2009). Ocean circulation kinetic energy: Reservoirs, sources and sinks. *Annual Review of Fluid Mechanics*, 41(1), 253–282. <https://doi.org/10.1146/annurev.fluid.40.111406.102139>
- Frezat, H., Le Sommer, J., Fablet, R., Balarac, G., & Lguensat, R. (2022). A posteriori learning for quasi-geostrophic turbulence parametrization. *Journal of Advances in Modeling Earth Systems*, 14(11). <https://doi.org/10.1029/2022ms003124>
- Friedlingstein, P., O'sullivan, M., Jones, M. W., Andrew, R. M., Gregor, L., Hauck, J., et al. (2022). Global carbon budget 2022. *Earth System Science Data*, 14(11), 4811–4900. <https://doi.org/10.5194/essd-14-4811-2022>
- Gauthier, J. (2014). Conditional generative adversarial nets for convolutional face generation. Class project for Stanford CS231N: Convolutional neural networks for visual recognition. Winter semester, 2014(5), 2.
- Gnanadesikan, A., Bianchi, D., & Pradal, M.-A. (2013). Critical role for mesoscale eddy diffusion in supplying oxygen to hypoxic ocean waters. *Geophysical Research Letters*, 40(19), 5194–5198. <https://doi.org/10.1002/grl.50998>
- Gnanadesikan, A., Pradal, M.-A., & Abernathey, R. (2015). Isopycnal mixing by mesoscale eddies significantly impacts oceanic anthropogenic carbon uptake. *Geophysical Research Letters*, 42(11), 4249–4255. <https://doi.org/10.1002/2015gl064100>
- Goodfellow, I. (2016). *Deep learning*. MIT press.
- Heuzé, C., Huhn, O., Walter, M., Sukhikh, N., Karam, S., Körtke, W., et al. (2023). A year of transient tracers (chlorofluorocarbon 12 and sulfur hexafluoride), noble gases (helium and neon), and tritium in the arctic ocean from the mosaic expedition (2019–2020). *Earth System Science Data*, 15(12), 5517–5534. <https://doi.org/10.5194/essd-15-5517-2023>
- Holloway, G. (1986). Estimation of oceanic eddy transports from satellite altimetry. *Nature*, 323(6085), 243–244. <https://doi.org/10.1038/323243a0>
- Huo, J., Zhang, J., Yang, J., Li, C., Liu, G., & Cui, W. (2024). High kinetic energy mesoscale eddy identification based on multi-task learning and multi-source data. *International Journal of Applied Earth Observation and Geoinformation*, 128, 103714. <https://doi.org/10.1016/j.jag.2024.103714>
- Jensen, T. G. (2001). Arabian Sea and Bay of Bengal exchange of salt and tracers in an ocean model. *Geophysical Research Letters*, 28(20), 3967–3970. <https://doi.org/10.1029/2001gl013422>
- Kingma, D. P. (2013). Auto-encoding variational Bayes. arXiv preprint arXiv:1312.6114.
- Kingma, D. P. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
- Klocker, A., & Abernathey, R. (2014). Global patterns of mesoscale eddy properties and diffusivities. *Journal of Physical Oceanography*, 44(3), 1030–1046. <https://doi.org/10.1175/jpo-d-13-0159.1>
- Klymak, J. M., Crawford, W., Alford, M. H., MacKinnon, J. A., & Pinkel, R. (2015). Along-isopycnal variability of spice in the north Pacific. *Journal of Geophysical Research: Oceans*, 120(3), 2287–2307. <https://doi.org/10.1002/2013jc009421>
- Kochkov, D., Yuval, J., Langmore, I., Norgaard, P., Smith, J., Mooers, G., et al. (2024). Neural general circulation models for weather and climate. *Nature*, 632(8027), 1060–1066. <https://doi.org/10.1038/s41586-024-07744-y>
- Kunze, E., & Smith, S. L. (2004). The role of small-scale topography in turbulent mixing of the global ocean. *Oceanography*, 17(1), 55–64. <https://doi.org/10.5670/oceanog.2004.67>
- LaCasce, J., Ferrari, R., Marshall, J., Tulloch, R., Balwada, D., & Speer, K. (2014). Float-derived isopycnal diffusivities in the dimes experiment. *Journal of Physical Oceanography*, 44(2), 764–780. <https://doi.org/10.1175/jpo-d-13-0175.1>
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436–444.
- Ledwell, J., Montgomery, E., Polzin, K., St. Laurent, L., Schmitt, R., & Toole, J. (2000). Evidence for enhanced mixing over rough topography in the abyssal ocean. *Nature*, 403(6766), 179–182. <https://doi.org/10.1038/35003164>
- Ledwell, J. R., Watson, A. J., & Law, C. S. (1993). Evidence for slow mixing across the pycnocline from an open-ocean tracer-release experiment. *Nature*, 364(6439), 701–703. <https://doi.org/10.1038/364701a0>
- Mahadevan, A. (2016). The impact of submesoscale physics on primary productivity of plankton. *Annual Review of Marine Science*, 8(1), 161–184. <https://doi.org/10.1146/annurev-marine-010814-015912>
- McWilliams, J. C. (2016). Submesoscale currents in the ocean. *Proceedings of the Royal Society, A*472(2189), 20160177. <https://doi.org/10.1098/rspa.2016.0117>
- Moum, J., Caldwell, D., Nash, J., & Gunderson, G. (2002). Observations of boundary mixing over the continental slope. *Journal of Physical Oceanography*, 32(7), 2113–2130. [https://doi.org/10.1175/1520-0485\(2002\)032<2113:oobmot>2.0.co;2](https://doi.org/10.1175/1520-0485(2002)032<2113:oobmot>2.0.co;2)
- Perezhogin, P., Zanna, L., & Fernandez-Granda, C. (2023). Generative data-driven approaches for stochastic subgrid parameterizations in an idealized ocean model. *Journal of Advances in Modeling Earth Systems*, 15(10), e2023MS003681. <https://doi.org/10.1029/2023ms003681>
- Radford, A. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks.
- Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F. A., et al. (2019). On the spectral bias of neural networks. In *International conference on machine learning* (pp. 5301–5310).
- Raimondi, L., Wefing, A.-M., & Casacuberta, N. (2024). Anthropogenic carbon in the Arctic Ocean: Perspectives from different transient tracers. *Journal of Geophysical Research: Oceans*, 129(1), e2023JC019999. <https://doi.org/10.1029/2023jc019999>
- Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378, 686–707. <https://doi.org/10.1016/j.jcp.2018.10.045>
- Richardson, P. L. (2008). On the history of meridional overturning circulation schematic diagrams. *Elsevier*, 76(4), 466–486. <https://doi.org/10.1016/j.pcean.2008.01.005>
- Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015: 18th International Conference, Munich, Germany, October 5–9, 2015, Proceedings, Part III* (Vol. 18, pp. 234–241). https://doi.org/10.1007/978-3-319-24574-4_28

- Shcherbina, A. Y., Sundermeyer, M. A., Kunze, E., D'Asaro, E., Badin, G., Birch, D., et al. (2015). The LatMix summer campaign: Submesoscale stirring in the upper ocean. *Bulletin America Meteorology Social*, 96(8), 1257–1279. <https://doi.org/10.1175/bams-d-14-00015.1>
- Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., & Woo, W.-c. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting (Vol. 28).
- Sirohi, M., & Thomas, J. (2024). Passive tracer dispersion by idealized flows across Rossby numbers. *Journal of Physical Oceanography*, 54(9), 1889–1902. <https://doi.org/10.1175/jpo-d-23-0132.1>
- Spiro Jaeger, G., MacKinnon, J. A., Lucas, A. J., Shroyer, E., Nash, J., Tandon, A., et al. (2020). How spice is stirred in the Bay of Bengal. *Journal of Physical Oceanography*, 50(9), 2669–2688. <https://doi.org/10.1175/jpo-d-19-0077.1>
- Su, H., Jiang, J., Wang, A., Zhuang, W., & Yan, X.-H. (2022). Subsurface temperature reconstruction for the global ocean from 1993 to 2020 using satellite observations and deep learning. *Remote Sensing*, 14(13), 3198. <https://doi.org/10.3390/rs14133198>
- Sundermeyer, M. A., Birch, D. A., Ledwell, J. R., Levine, M. D., Pierce, S. D., & Cervantes, B. T. K. (2020). Dispersion in the open ocean seasonal pycnocline at scales of 1–10 km and 1–6 days. *Journal of Physical Oceanography*, 49(2), 415–437. <https://doi.org/10.1175/jpo-d-19-0019.1>
- Taylor, J. R., & Thompson, A. F. (2023). Submesoscale dynamics in the upper ocean. *Annual Review of Fluid Mechanics*, 55(1), 103–127. <https://doi.org/10.1146/annurev-fluid-031422-095147>
- Thomas, J. (2023). Turbulent wave-balance exchanges in the ocean. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 479(2276), 20220565. <https://doi.org/10.1098/rspa.2022.0565>
- Thomas, J., & Vishnu, R. (2022). Turbulent transition of a flow from small to $O(1)$ Rossby numbers. *Journal of Physical Oceanography*, 52(11), 2609–2625. <https://doi.org/10.1175/jpo-d-21-0270.1>
- Thompson, A. F., Lazar, A., Buckingham, C. E., Naveira Garabato, A. C., Damerell, G. M., & Heywood, K. J. (2016). Open-ocean submesoscale motions: A full seasonal cycle of mixed layer instabilities from gliders. *Journal of Physical Oceanography*, 46(4), 1285–1307. <https://doi.org/10.1175/jpo-d-15-0170.1>
- Wang, H., Grisouard, N., Salehipour, H., Nuz, A., Poon, M., & Ponte, A. L. (2022). A deep learning approach to extract internal tides scattered by geostrophic turbulence. *Geophysical Research Letters*, 49(11), e2022GL099400. <https://doi.org/10.1029/2022gl099400>
- Wang, R., Kashinath, K., Mustafa, M., Albert, A., & Yu, R. (2020). Towards physics-informed deep learning for turbulent flow prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 1457–1466).
- Warren, B. A., & Wunsch, C. (1981). Evolution of physical oceanography: Scientific surveys in honor of henry stommel (Vol. 664).
- Wunsch, C. (1970). On oceanic boundary mixing. In *Deep Sea Research and Oceanographic Abstracts* (Vol. 17, pp. 293–301). [https://doi.org/10.1016/0011-7471\(70\)90022-7](https://doi.org/10.1016/0011-7471(70)90022-7)
- Wunsch, C. (1997). The vertical partition of oceanic horizontal kinetic energy and the spectrum of global variability. *Journal of Physical Oceanography*, 27(8), 1770–1794. [https://doi.org/10.1175/1520-0485\(1997\)027<1770:tvpooh>2.0.co;2](https://doi.org/10.1175/1520-0485(1997)027<1770:tvpooh>2.0.co;2)
- Yu, X., Naveira Garabato, A. C., Martin, A. P., Buckingham, C. E., Brannigan, L., & Su, Z. (2019). An annual cycle of submesoscale vertical flow and restratification in the upper ocean. *Journal of Physical Oceanography*, 46(6), 1439–1461. <https://doi.org/10.1175/jpo-d-18-0253.1>
- Zhang, G., Chen, R., Li, X., Li, L., Wei, H., & Guan, W. (2023). Temporal variability of global surface eddy diffusivities: Estimates and machine learning prediction. *Journal of Physical Oceanography*, 53(7), 1711–1730. <https://doi.org/10.1175/jpo-d-22-0251.1>
- Zhang, Z., & Sabuncu, M. (2018). Generalized cross entropy loss for training deep neural networks with noisy labels. *Advances in Neural Information Processing Systems*, 31.
- Zhurbas, V., & Oh, I. S. (2004). Drifter-derived maps of lateral diffusivity in the Pacific and Atlantic Oceans in relation to surface circulation patterns. *Journal of Geophysical Research*, 109(C5). <https://doi.org/10.1029/2003jc002241>