Ocean Modelling 156 (2020) 101707

Contents lists available at ScienceDirect

Ocean Modelling

journal homepage: www.elsevier.com/locate/ocemod

Predicting particle trajectories in oceanic flows using artificial neural networks

Matthew D. Grossi [a,*], Miroslav Kubat [b], Tamay M. Özgökmen [a]

[a] Rosenstiel School of Marine and Atmospheric Science, University of Miami, FL, United States of America
[b] Department of Electrical and Computer Engineering, University of Miami, FL, United States of America

ARTICLE INFO

Keywords:
Advection
Current velocity
Mass transport
Particle motion
Current prediction
Machine learning

ABSTRACT

Predicting ocean transport has many practical applications ranging from search and rescue operations to predicting the spread of oil, debris, and biogeochemical tracers, yet trajectory prediction remains a challenge for existing ocean modeling techniques. General circulation models require high resolution observational data in order to be properly initialized, but these data do not exist for the ocean. Statistical models are difficult to tune with existing data and are often too simple to accurately encapsulate turbulent flows. Here we investigate a data-driven approach to ocean transport prediction wherein the goal is to first learn from available data instead of prescribed laws of physics and then apply this information to new data. More specifically, we explore whether simple artificial neural networks (ANNs) are capable of learning to predict 2D particle trajectories using only previous velocity observations. ANNs are trained in two ways: first, a so-called "one-to-one ANN" uses a particle's most recently observed velocity to predict its velocity six hours later, and second, a "time series ANN" uses the past 24 hours' worth of velocity observations to predict the next 24 h. We present a proof-of-concept considering particles in a hierarchy of simulated flow regimes ranging from uniform, steady flow to more complex scenarios with interacting scales of motion and then substantiate our approach on trajectories in modeled flows generated by a high-resolution Hybrid Coordinate Ocean Model for a mesoscale eddy in the northern Gulf of Mexico. We also assess ANN sensitivity to the prediction window over which forecasts are made, the number of training particles used, and the size of the network. ANNs successfully predict 24 h trajectories within the temporal bounds of the training data with forecast errors around half those of both rudimentary persistence and classical ARIMA models. Predicting beyond the domain of the training data leads to forecast errors comparable to ARIMA models. Our results suggest that ANNs offer promising potential as a data-driven approach to forecasting material transport in the ocean.

1. Introduction

Efforts to predict material transport in a turbulent ocean are motivated by many socioeconomic interests including oil spill response (Poje et al., 2014; Özgökmen et al., 2016), search and rescue operations (Isaji et al., 2006), and anticipating the spread of harmful algae blooms, pollutants, and marine debris (Enriquez et al., 2010; Olascoaga, 2010; Olascoaga and Haller, 2012; Normile, 2014). Traditional ocean forecasting techniques merge theoretical knowledge of the underlying flow dynamics with information about the initial state from which the system evolves. "Ground truth" data are now widely available via real-time ocean observing initiatives from both Lagrangian and Eulerian perspectives. These data include direct *in situ* velocity measurements such as from current meters and drifters (Shay et al., 1998; Lumpkin et al., 2017; Novelli et al., 2018) along with products derived from passive monitoring systems such as high-frequency radar and satellite altimetry (Barrick et al., 1974, 1977; McGoogan, 1975; Roarty et al., 2010).

Despite decades of advances in observational and computational methods, conventional theory-driven ocean forecasting tools continue to be inhibited by the prevailing sparsity of ocean data (Özgökmen et al., 2009; Bolton and Zanna, 2019). One problem is that not enough data exist to properly initialize predictive models. The highest resolution ocean general circulation models (OGCMs) numerically compute Eulerian velocity at $\mathcal{O}(10^9)$ spatial points at any given time (Özgökmen et al., 2009), each of which requires unique initial conditions (*e.g.*, Mariano et al., 2011; Wei et al., 2016). Still, the ocean remains under sampled to such a degree that even coarse resolution OGCMs are under constrained by available data, especially outside of coastal regions. While an OGCM may be initialized with low resolution climatology in the absence of observations, this seldom provides suitable starting conditions for very localized problems such as particle prediction.

* Corresponding author.
E-mail addresses: matthew.grossi@rsmas.miami.edu (M.D. Grossi), mkubat@miami.edu (M. Kubat), tozgokmen@rsmas.miami.edu (T.M. Özgökmen).

https://doi.org/10.1016/j.ocemod.2020.101707
Received 19 November 2019; Received in revised form 11 August 2020; Accepted 30 September 2020
Available online 13 October 2020

Two examples illustrate the sensitivity of the particle prediction problem to accurate initial conditions. First, even in simple laminar flows, two identical Lagrangian drifters released simultaneously side-by-side eventually diverge and travel very different routes due to the unsteady time-dependence of oceanic flow (*e.g.*, Aref, 1984; Yang and Liu, 1997; Özgökmen et al., 2001; Koshel' and Prants, 2006). This chaotic particle behavior is notoriously difficult to predict because even slight differences in starting conditions yield vastly different forecasts. A second example is the case of a drifter encountering two adjacent counter-rotating eddies. The systems of streamlines that encircle the centers of rotation form so-called Lagrangian coherent structures (LCS) that are known to control material transport (Peacock and Haller, 2013; Olascoaga et al., 2013). Whether this drifter gets drawn into the inward flow of one eddy or repelled by the outward flow of the second eddy depends strongly on its launch position relative to the two eddies and to the singular streamline that divides them (Molcard et al., 2006). Small errors in the initial location of either the drifter or the critical streamline can again result in forecasts that are drastically different from actual observations.

A second challenge with particle prediction is the inability to adequately sample the complexity of ocean dynamics. Locating critical regions around LCSs in nature is nearly impossible (Haza et al., 2007) because observing them requires impractically high numbers – *i.e.*, millions – of drifters. Ocean forcing mechanisms cause unique and independent dynamic features to exist across a wide range of scales (Stommel, 1948; Holland and Lin, 1975a,b). Fully capturing these ocean dynamics requires that a minimum of $\mathcal{O}(10^{23})$ spatial points be sampled simultaneously (using a typical Reynold's number $R_e \approx 10^{10}$; Özgökmen et al., 2009). Even the finest ocean experiments, which might sample thousands of spatial points, do not come close to this requirement. Consequently, the complexity of relevant steering dynamics on all scales of motion is neither fully understood nor accurately replicated in predictive OGCMs.

Of particular importance to material transport are localized submesoscale dynamics, characterized by spatial scales <10 km and time scales on the order of hours to days. These dynamics have been found to significantly impact particle trajectories in the ocean everywhere from the Gulf of Mexico (*e.g.*, Poje et al., 2014; Berta et al., 2016) to the Beaufort Sea (Mensa et al., 2018). Unlike mesoscale dynamics, submesoscales are strongly influenced by vertical motions (*e.g.*, Smith et al., 2015; Choi et al., 2017; Mensa et al., 2018). Surface convergence fronts are common at these scales and function dualistically as across-front barriers to transport and as along-front transport "highways" (Schroeder et al., 2012; Smith et al., 2015; Kuitenbrouwer et al., 2018; Taylor, 2018). Haza et al. (2016) also identified submesoscale "leakage" pathways by which particles may escape larger mesoscale circulations. While mesoscales are usually fully resolved by OGCMs, submesocale dynamics are often parameterized due to computational restrictions on model resolution. Even if partially resolved, the modeled submesoscale dynamics are not always physically realistic because constraining them is difficult with available observations. This ultimately leads to discrepancies between numerical simulations and real observations (Dueben and Bauer, 2018).

Stochastic or Markovian models provide an alternative to calculating velocity at large numbers of grid points by considering flow fields statistically. When used in conjunction with OGCMs, unresolved physics are added to the statistical mean flow as random increments. Zero-th, first, or second order Markovian models calculate successive positions, velocities, or accelerations, respectively. Griffa (1996) provided the first oceanographic evaluation of stochastic models by using constant parameters corresponding to known time and length scales of the mean flow and showed that particle motion in the upper ocean can be closely simulated using first-order Markovian models with a finite turbulent velocity scale. Berloff and McWilliams (2003) developed these further to better replicate interactive mesoscale ocean dynamics by randomizing the model parameters. This allowed particles to also move between features, as is commonly observed in nature (Berta et al., 2016; Haza et al., 2016). Haza et al. (2012) later merged statistical subgrid models with mesoscale LCSs computed from OGCMs to better simulate particle behavior on submesoscales.

Markovian models are among the cheapest trajectory models to run: with everything parameterized, trajectories can effectively be generated in a matter of seconds. Nevertheless, it is often difficult to determine model parameters from available data, and tuning the parameters once the form of the differential equations has been set can be tedious. While targeted sampling of multi-scale dynamics, such as in the Grand Lagrangian Deployment (GLAD) experiment in the northern Gulf of Mexico, have provided new insights into ocean velocity statistics, they have also highlighted performance shortcomings in existing statistical prediction methods (Beron-Vera and LaCasce, 2016; Mariano et al., 2016).

To summarize, while traditional theory-driven ocean models are excellent tools for better understanding fluid dynamics and for exploring new phenomena, they struggle as predictive tools because (1) the sparsity of observational data makes it impossible to accurately initialize high resolution ocean models, (2) not enough data exist to observe complete ocean dynamics in nature nor to properly replicate them in OGCMs, and (3) important steering dynamics at local submesoscales are often poorly captured by OGCMs due to computation restraints on model resolution. Initialization and tuning challenges also affect simpler statistical models that do not rely on solving primitive equations at millions of grid points. In light of these challenges, an alternative approach to ocean transport prediction may be one that starts with objective observations instead of prescribed laws of physics.

Here we take initial steps of exploring this possibility in the field of physical oceanography by testing whether ocean particle trajectories contain inherent predictability that a data-driven algorithm might "discover". Our goal, broadly speaking, is to apply information learned from past data to predict future states. Data-driven strategies such as this form the foundation of machine learning techniques that have become enormously popular for tackling complex problems in 21st-century data science. We now present the particle prediction problem more specifically within the framework of data-driven machine learning.

2. Data-driven machine learning for ocean prediction

Data-driven modeling is built upon the rationale that cause and effect relationships are often discernible among relevant variables and observed outcomes (Solomatine et al., 2008). For example, one intuitively expects a relationship between wind speed and sea state. Data-driven models are trained to identify statistical relationships between data sets and later use this "learned" information to make predictions about cases unseen during training. Although these models do not explicitly provide insight into the underlying physics of a problem, they are useful in situations when knowing an outcome is more important than understanding the relevant physical mechanisms. Oil spill first responders, for instance, are more concerned about where oil is and where it is going than why it moves as it does. Data-driven models can also be beneficial when the full physics of a problem are not known, when the underlying physics are not easily modeled numerically, or when large quantities of relevant data are available from which to learn (Solomatine et al., 2008).

Considerable effort has been made over several decades in fluid dynamics to use data-driven machine learning as alternatives to the computationally expensive task of solving the nonlinear Navier–Stokes equations. One increasingly popular method attempts to learn governing partial differential equations from large quantities of numerical model output or experimental data. This approach starts with observations of one or more state variables and their time derivatives, assembles a library of nonlinear features in the form of polynomial functions, and finally identifies the relevant components in the feature

library using weights obtained by a machine learning algorithm (*e.g.*, Brunton et al., 2016; Ayed et al., 2019; Ouala et al., 2019). The discovered equations are then integrated in time to forecast future system states. Other work has focused on learning flow dynamics from snapshot images (*e.g.*, Lee and You, 2017; Mohan and Gaitonde, 2018). Still others develop sophisticated black box machine or deep learning methods to either combine with or altogether bypass conventional models (Chen et al., 2018; Wikner et al., 2020). These and similar studies in a rapidly growing body of literature boast promising potential for learning and predicting complex flow dynamics, but most depend on having observations at resolutions and densities not yet available to the oceanographer.

An important aspiration of the present investigation is to replace the computationally expensive process of running circulation models with information learned entirely from available observations. This requirement necessarily precludes methods that rely on high resolution model output for training. Perhaps the easiest instruments for the oceanographer to deploy in relation to this problem are Lagrangian surface drifters that provide time series of position, velocity, and acceleration throughout a region of interest. Given the nature of the problem and the hypothetical data, we chose an artificial neural network (ANN), a biologically-inspired data-driven nonlinear regression model whose notable strength is its theoretical ability to fit any function. Neural networks have demonstrated promising potential in a variety of complex geophysical problems such as identifying climate feedback mechanisms (González et al., 2015), weather forecasting (Dueben and Bauer, 2018), predicting climate patterns (Toms et al., 2020), and hurricane track prediction (Moradi Kordmahalleh et al., 2016). In oceanography, neural networks have been developed for locating spilled oil in synthetic aperture radar imagery (Kubat et al., 1998), for increasing the efficiency of parameterization routines in numerical ocean models (Krasnopolsky et al., 2002), for modeling nonlinear wind-wave spectra (Tolman et al., 2005), and for ocean eddy identification and tracking (Franz et al., 2018; Lguensat et al., 2018).

Here we investigate whether simple ANNs can learn to use observed velocities of drifting Lagrangian particles to predict future velocities. Our study differs from others in the oceanographic literature that explore machine learning tools in that we develop from scratch and test the simplest possible ANN instead of using more sophisticated forms of neural networks from preexisting software packages. Our rationale behind using a simple feedforward ANN is twofold. First is the law of parsimony, credited to the fourteenth century philosopher William of Ockham and supported by modern probability theory, which states that given two explanations for some phenomenon, the simpler one is more likely to be correct. In light of this, and following the inspiration of influential pioneering work in neural networks (*e.g.*, Elman, 1993), we choose to start small and simple.

The second reason for choosing a simple feed-forward network is more practical. The decades-old "black box" objection to neural networks is that their performance "under the hood" – or within a chosen machine learning library – is physically uninterpretable (see, *e.g.*, Touretzky and Pomerleau, 1989). Open research questions within the machine learning field related to this include: What exactly did the neural network learn, and how is it solving this particular problem? What do the weights signify, what do the neuron activation behaviors mean, and what are the patterns learned within the network? While these questions are well beyond the scope of the present investigation, answers would be of great interest to the oceanography community accustomed to extracting direct physical interpretations from theory-driven models. One can therefore argue that the more complex the machine learning architecture, the more physically abstract its behavior is likely to be and the lower the likelihood that physical interpretability will ever be identified. Sometimes simpler is better.

The success of any data-driven method fundamentally requires some degree of predictability among the desired output data (*e.g.*, Massoud et al., 2018). The predictability of ocean Lagrangian trajectories has been extensively explored both numerically (*e.g.,* Özgökmen et al., 2000; Rixen et al., 2008) and experimentally (*e.g.,* Özgökmen et al., 2001; Coelho et al., 2015). Özgökmen et al. (2000, 2001) use simple statistical models that closely resemble ANNs to show how trajectory prediction on the order of days to weeks can be improved by combining information from nearby Lagrangian drifters with knowledge of the mean flow within a region of interest. One then wonders whether Lagrangian drifter data alone contain enough information from which to predict trajectories. If so, then an ANN ought to be able to learn from Lagrangian trajectories without any additional information about the underlying physics. This study takes the first step toward answering this question by systematically exploring the inherent predictability of ocean trajectories using simple artificial neural networks.

A simple test of predictability of a time series is whether or not future states can be predicted given only information about previous states. For example, the sequence (2, 4, 6, 8, 10, x) contains enough information for one to predict the value of x to be 12. Similarly, one can test whether particle trajectories contain inherent predictability by trying to predict where a particle will go next based on where it has already been. This is the premise of all regression techniques and is the logical starting point for testing any new time series prediction scheme.

An early non-oceanographic demonstration of this technique in an industrial robotic application was by Payeur et al. (1995) who designed neural networks to predict position, velocity, and acceleration of moving objects based on previous observations. This work suggests that the sequential velocities of drifting ocean particles may be learned in a similar fashion. We therefore pose several questions: *Can a neural network be trained to predict a particle's future velocities using only its previous velocities as input? If so, how would its predictive skill compare to a more traditional autoregressive model? If this predictive task proves challenging, can the ANN at least perform better than a rudimentary persistence model?*

The first part of this study consists of a series of proof-of-concept control experiments involving particles in a hierarchy of simulated known flows. Deliberately engineering the flow field allows us to gradually "turn up" the complexity of the dynamics – which inversely decreases the predictability of Lagrangian trajectories – in order to systematically determine where predictability collapses. Of the many tested flow fields, three are presented here. We then conduct a case study in physically-based oceanic flow fields generated by the HYbrid Coordinate Ocean Model (HYCOM) for a mesoscale eddy in the Gulf of Mexico to assess how our approach might perform with observational data in a real-ocean application.

Two types of ANNs are developed: one predicts a particle's velocity at some future time based only on its previous velocity, and another uses as input one day's worth of observations. ANNs are evaluated against predictions from both rudimentary persistence and autoregressive integrated moving average (ARIMA) models. We find that ANNs predict trajectories of particles not seen during training, often with half the forecast error of ARIMA. We also consider ANN sensitivity to the numbers of hidden layers, hidden neurons, and training particles, and to the prediction window over which forecasts are made. Importantly, our methodology is easily applied to field applications by replacing simulated particle velocities with those from Lagrangian drifters anywhere in the ocean. Our results suggest that data-driven ANNs may offer a missing supplement to traditional theory-driven models by providing forecasting tools that learn entirely from objective ground truth observations.

3. Proof-of-concept

3.1. Methods

We first conduct a hierarchy of control experiments consisting of buoyant particles in simulated rotational coherent structures of increasing complexity:

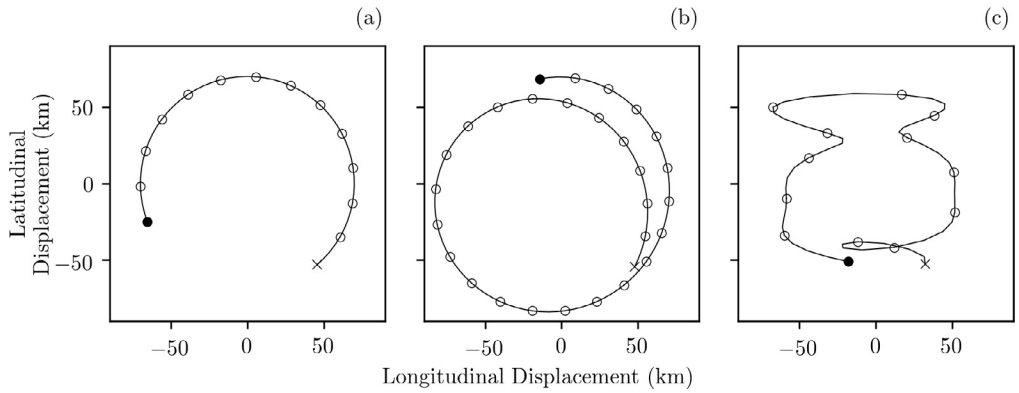

Fig. 1. Representative particle trajectories for control cases 1, 2, and 3: steady, uniform, stationary eddy (a); steady, uniform, drifting eddy (b); drifting eddy with time-dependent pulses (c). Trajectories start at solid circles and terminate at the "×". Open circles indicate the start of a new day.

1. Steady, uniform, and stationary eddy (center of rotation is not moving within the domain);
2. Steady, uniform, and drifting eddy (center of rotation moves in time); and
3. Drifting eddy with time-dependent sinusoidal pulses of amplitude a and frequency ω added to the mean rotational flow.

Each flow field is scaled according to a realistic mesoscale eddy in the Gulf of Mexico studied by Haza et al. (2016) as a known example of interacting mesoscale and submesoscale motions. A description of the parameter space for each flow field follows.

3.1.1. Case 1: Stationary mesoscale eddy

If Eulerian velocity $\mathbf{u}$ as a function of space $\mathbf{x} = (x, y)$ and time t is separated into a time-independent mean circulation $\bar{u}$ and a time-dependent fluctuating component u':

$$\mathbf{u}(\mathbf{x}, t) = \bar{u}(\mathbf{x}) + u'(\mathbf{x}, t) \tag{1}$$

then Case 1 represents mean eddy flow only with no small scale turbulence; that is, $u' = 0$ and $\mathbf{u}(\mathbf{x}, t) = \bar{u}(\mathbf{x})$. Velocity components for circular mean flow are given by:

$$\bar{u}(x, y) = \frac{2C_o(y - y_c)}{r^2} \exp\left[\frac{-(x - x_c)^2 - (y - y_c)^2}{r^2}\right] \tag{2a}$$

$$\bar{v}(x, y) = -\frac{2C_o(x - x_c)}{r^2} \exp\left[\frac{-(x - x_c)^2 - (y - y_c)^2}{r^2}\right] \tag{2b}$$

where (x, y) are the zonal and meridional positions of the particle, respectively; (x_c, y_c) is the center of the eddy, taken as the Cartesian origin for convenience; $r = 70 \times 10^3$ m is the radius at which the flow velocity is a characteristic mean velocity $\tilde{u} = \tilde{v} = 0.1\,\mathrm{m\,s^{-1}}$; and $C_o \approx 2.6 \times 10^4\,\mathrm{m^2\,s^{-1}}$ is a dimensional parameter that sets the mean velocity $(\tilde{u}, \tilde{v})$ at radius r, calculated using Eq. (2) with $x = y = r$ and $x_c = y_c = 0$ and where $C_o > 0$ produces clockwise flow. Given the absence of temporal flow variance, particles released in Case 1 will travel around a perfect circle. A representative particle trajectory is shown in Fig. 1a, starting at the solid circle and ending at the "×".

3.1.2. Case 2: Drifting mesoscale eddy

Case 2 is set up identically to Case 1 except that the center of the eddy $\mathbf{x}_c = (x_c, y_c)$ varies in time:

$$\mathbf{x}_c = mt \tag{3}$$

where $m = -8 \times 10^{-3}\,\mathrm{m\,s^{-1}}$ to generate a steady southwesterly drift of approximately $0.7\,\mathrm{km\,d^{-1}}$, comparable to the observed eddy drift in Haza et al. (2016) over the course of ≈14 days. Particles in this second case traverse a 2D spiral pattern over time as seen in Fig. 1b.

3.1.3. Case 3: Drifting mesoscale eddy with submesoscale noise

Case 3 introduces time-dependent velocity fluctuations to the drifting eddy of Case 2 in the form of periodic pulses with amplitude $a = 1.5\,\mathrm{m\,s^{-1}}$, frequency $\omega = 2\pi/T$, and period $T = 4$ h:

$$u' = a\sin(-\omega t) - 2a\cos(-1.5\omega t) \tag{4a}$$

$$v' = a\cos(-\omega t) - 2a\sin(-1.5\omega t) \tag{4b}$$

Adding four unique pulse signals to the mean flow (Eq. (4)) simulates the combined signals of multiple coinciding forcing mechanisms that one expects in nature. The resulting flow pattern (Fig. 1c) can be likened to a mesoscale eddy surrounded by smaller submesoscale circulations. It is worth noting that in this simulation, all particles remain trapped within the mesoscale circulation, though this is seldom true in nature where scales are dynamically interconnected (see Haza et al., 2016). A more realistic scenario of interacting scales of motion will be considered in the case study described in Section 4.

Sensitivity studies have demonstrated that a minimum of $\mathcal{O}(200 - 400)$ drifters are necessary to measure and quantify mesoscale–submesoscale interaction dynamics (Özgökmen et al., 2011, 2012). The strong dependence of local transport on submesoscales dictates that real-ocean applications would ideally involve at least $\mathcal{O}(300)$ drifters deployed within $\mathcal{O}(1 - 10)$ km region. We therefore chose to release 360 particles in each simulation. This allows a neural network to be trained using 270 (or 75%; see Section 3.1.4) of them, within the threshold for capturing scale interactions. We later consider the inevitable practical challenges of deploying such quantities of drifters by decreasing the number of particles to more operationally realistic quantities in the eddy case study.

In all three cases, particles are evenly distributed around a circle of radius 70×10^3 m from the center of the eddy and released simultaneously. They are advected for 14 days in Cases 1 and 3 and for 29 days in Case 2 in order to capture the spiral trajectory effect by making a complete revolution around the eddy. The trajectories of these particles – hereafter, the real trajectories – are determined by applying a fourth-order Runge–Kutta (RK4) integrator (Dormand and Prince, 1980) to Eq. (2) with data sampling frequency $\Delta_s t$. We choose $\Delta_s t = 3$ h to be comparable to the frequency of the HYCOM output used in the case study described in Section 4. Velocity components (u_1, v_1) at each time step are found by substituting new positions (x_1, y_1) into Eq. (2).

3.1.4. Neural network configurations

We now describe the setup of our artificial neural networks relative to the data and to the problem at hand. Specifics of the neural network architectures, development, and training process are detailed in Appendix.

We engineer ANNs to predict a particle's future velocity at some time $t_o + \Delta_p t$, where subscript p indicates a prediction time window. Thus, for every ANN input $\mathbf{v}_{t_o}$, the target output is $\mathbf{v}_{t_o + \Delta_p t}$, and each

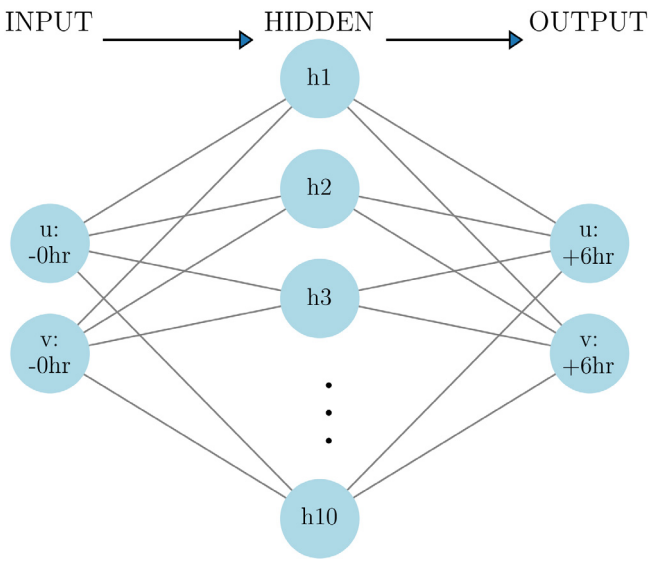

Fig. 2. One-to-one neural network containing two input neurons, ten hidden neurons arranged in a single hidden layer, and two output neurons. In simple feed-forward networks, adjacent layers are fully inter-connected by weighted links (lines) but neurons within the same layer are completely independent of each other. Velocity $\mathbf{V}_{t_o} = (u_{t_o}, v_{t_o})$ is fed through the network to produce the output vector $\mathbf{V}_{t_o+\Delta t} = (u_{t_o+\Delta t}, v_{t_o+\Delta t})$.

$(\mathbf{v}_{t_o}, \mathbf{v}_{t_o+\Delta_p t})$ pair constitutes a unique example. The choice of $\Delta_p t$ is based more on engineering considerations than on physics. If the time series is predictable, the ANN will learn whatever time interval it is trained with. Longer prediction time windows lead to flatter error curves over a forecast period, but at decreased forecast resolution. Such decisions are highly problem-specific, but allow for many possibilities, such as creating a system of ANNs with some making longer forecasts while others are trained to make shorter forecasts.

We choose $\Delta_p t = 6\,\mathrm{h}$ for two reasons. First, having a different prediction window than the observational time step (*i.e.*, $\Delta_p t \neq \Delta_s t$) provides a better sense of how well the network learns the data. Instead of simply predicting the next time step, the network must predict two steps into the future. The second reason relates to an error propagation problem when making forecasts beyond $\Delta_p t$. In this configuration, a 24-h forecast would be made recursively starting with the most recent velocity observations followed by the successive outputs:

$$h_{(24\,\mathrm{h})}(x) = h_{(18\,\mathrm{h})}\left(h_{(12\,\mathrm{h})}\left(h_{(6\,\mathrm{h})}(\mathbf{v}_{(0\,\mathrm{h})})\right)\right) \tag{5}$$

where $h(x)$ represents the ANN's hypothesis for the next velocities. We seek to minimize this error propagation while still maintaining a prediction window that is small enough to be practical in applications such as oil spill forecasting. We refer to ANNs configured in this manner as *one-to-one ANNs* because one observation is used to predict one future state.

If velocity $\mathbf{v}$ is decomposed into zonal and meridional components (u, v), then each observation becomes a vector containing two attributes. Each attribute is assigned to an input and an output neuron as shown in Fig. 2. Network weights (lines in Fig. 2) are initiated randomly from a univariate Gaussian distribution of mean 0 and variance 1.

Due to the time dependence and high variability of ocean dynamics, we combine a floating window approach to training (first proposed by Kubat, 1989) with a continuous or so-called online learning paradigm as follows. Of the 360 particles in each experiment, 75% or 270 are used for training and the remaining 25% are reserved for testing. The size of the training data set is fixed to include only the most recent 24 hours' worth of observations. With 270 training particles and $24/3 = 8$ observations/day, this amounts to 2160 training examples at any given time. Observations are weighed at each training step according to how recent they are by assigning to the oldest 12 hours' worth of observations a weight of 1, to the next 6 h a weight of 2, and to the most recent 6 h a weight of 3, where the weight indicates how many times the example is duplicated in the training set. Weighing the observations introduces repetition to the training process and increases the number of training examples to 3780.

The network is initially trained for 100 epochs using the observations from Day 1, where one epoch indicates one round through every training example. Subsequent observations from each particle are then added to the training set one time step $\Delta_s t$ at a time to mimic, for example, a Lagrangian drifter reporting its position every n hours. Old examples are discarded as new data become available and training continues for another 100 epochs each time the training set is updated. This happens a total of 104 ($13\,\mathrm{d} \times 8$ obs/d) times for Cases 1 and 3 (224 times for Case 2); thus, by the end of the simulation, the ANNs have been trained for a total of either 105,000 or 225,000 epochs.

Such extensive training greatly increases the risk of overfitting. To combat this, we recognize that each training epoch essentially produces a new version of the ANN. We therefore select the best model (out of 100) at the end of every training session (see Appendix). This allows for three possible scenarios after updating the training set throughout the simulation:

1. If the model benefited from all additional training, the final version of the model after epoch 100 is retained.
2. If training the model for an additional 100 epochs causes the model to overfit such that it performed better before the additional training, the original version of the model is retained.
3. If the model benefited from some additional training but began to overfit during the training session, the version of the model just before overfitting began is retained.

Using an ANN with the fewest possible hidden neurons also aids against overfitting, since the regression flexibility of an ANN is proportional to the number of hidden neurons.

In addition to the one-to-one networks, we also develop *time series ANNs* that take as input a time series of data rather than a single observation. This better reflects the physical reality that a particle's next location can often be anticipated from its most recent n positions (or velocities). Under this configuration, each particle constitutes an individual training example containing 16 attributes (8 observations per day × 2 velocity components). This results in a tenfold decrease in the number of training examples available (*i.e.*, from 2160 to 270) at any given time. The time series is not weighted for these ANNs because doing so would increase the number of attributes per example rather than the number of examples. Thus, it does not introduce the same repetition as it does with the one-to-one networks. Instead, it would require an unnecessarily larger network with more input and hidden neurons.

While extended forecasts can be made recursively with one-to-one ANNs using Eq. (5), this is not so straightforward with time series ANNs when $\Delta_p t \neq \Delta_s t$. These ANNs are engineered to expect the input vector to contain observations with time step $\Delta_s t$. If, after training, a vector containing a different time step is fed through the network, the output will be nonsensical. Thus, in the case where $\Delta_p t > \Delta_s t$, if the ANN predicts a single future time step $\mathbf{v}_{t_o+\Delta_p t}$, then one would need to interpolate between the predicted velocity and the most recent input velocity in order to update the input time series and feed it back to the ANN. Even more problematically, if $\Delta_p t < \Delta_s t$ and the ANN predicts only one future time step, extended forecasts are altogether impossible without extrapolation. A solution to this problem is to engineer the time series ANN to predict multiple time steps simultaneously, where both $\Delta_p t$ and the number of time steps to predict are determined by the specific problem. We therefore engineer our time series ANNs to issue a 24-hr forecast as a set of velocities at $t_o + i\Delta_p t$, where $\Delta_p t = 6\,\mathrm{h}$ and $i \in (1, 2, 3, 4)$, as illustrated in Fig. 3.

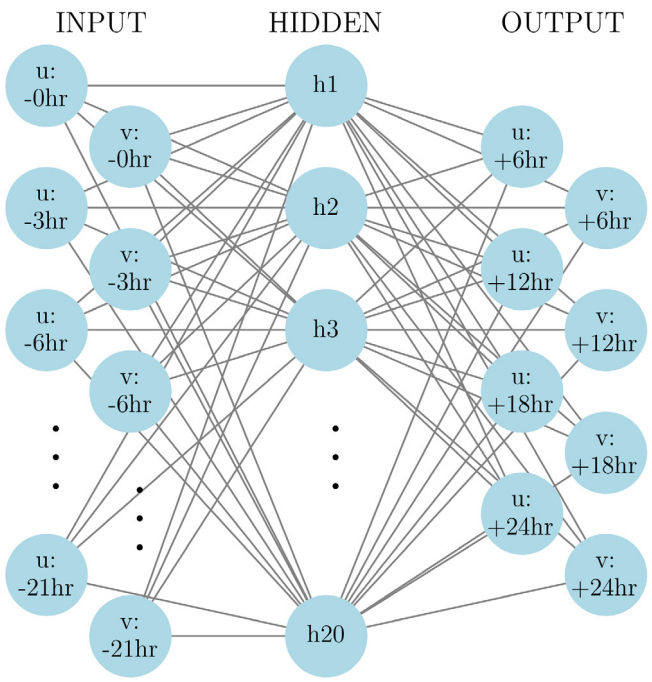

Fig. 3. Time series neural network containing 16 input neurons, 20 hidden neurons in a single hidden layer, and 8 output neurons. Input vector $\mathbf{V}_{\text{in}} = (u_{t_o}, \ldots, u_{t_0-n}, v_{t_o}, \ldots, v_{t_o-n})$, where n is the number of time steps back, and output vector $\mathbf{V}_{\text{out}} = (u_{t_o+6\text{hr}}, \ldots, u_{t_0+24\text{hr}}, v_{t_o+6\text{hr}}, \ldots, v_{t_o+24\text{hr}})$ in 6-h increments.

The learning rate η, an increment that controls network weight adjustment during training (see Appendix), is set to 10^{-1} for one-to-one and time series ANNs in Cases 1 and 2. Trial and error indicated that the complexity of Case 3 requires slower learning, so $\eta = 10^{-2}$ for both network configurations. All ANNs contain a single hidden layer for simplicity. For the one-to-one ANNs, where each example is composed of two attributes, we use 10 hidden neurons, while for the time series ANNs we use 20 hidden neurons due to the larger number of attributes per example.

Because ANNs are initialized randomly, a network may occasionally learn faster or slower depending on where in solution space the initial weights are located. We account for this by randomly initializing an ensemble of three unique ANNs of both configurations for each experiment. Thus, a total of 18 different ANNs are trained and tested during the three proof-of-concept simulations.

All particles are assumed to be released at 00:00 on Day 1. As the particles circle the eddy, 24-h forecasts for all 90 test particles are issued every midnight (starting on day 2) using the most recent version of the network. Hereafter, we refer to the times at which forecasts are initiated as forecast times and the times for which velocities are predicted (*i.e.*, 06:00, 12:00, 18:00, and 24:00) as prediction times. Thus, each trial produces a total of 13 forecasts in Cases 1 and 3 and 28 forecasts in Case 2, where each forecast includes four predictions for 90 particles.

3.1.5. Assessment metrics

The most fundamental assessment metric for any prediction model is rudimentary persistence, a mere continuation in both space and time of the most recent observation. It is commonly used because of its simplicity, but it also sets the lowest possible standard. Because ocean particle prediction is a notoriously difficult problem, it is logical to set low expectations, at least at first. Thus, we first compare ANN prediction errors to those of persistence predictions in each experiment.

For a far more rigorous metric, we employ statistical autoregressive integrated moving average (ARIMA) models that make predictions based on linear combinations of time series lags and the lagged forecast errors. These models contain three parameters that must be tuned for any given time series. First, the autoregression parameter p indicates the number of lags to be included in the model and is determined by the correlations between an observation and each previous observation. An integrated or differencing technique subtracts an observation from the previous observation in an attempt at making the time series stationary. A parameter d specifies how many times this differencing is performed. Finally, the moving average parameter q specifies the number of lags to be used in a moving window average that is compared to the forecast residual error.

In most time series applications, the optimal combinations of p, d, and q are determined by brute force trial and error. Here, however, with $\mathcal{O}(10^4)$ time series to fit, this is hardly feasible. We instead use a parameter search function *auto.arima* in **R** (from "forecast" package, **R** 3.4.2, using default arguments) to fit a unique ARIMA model to normalized u and v time series for each of the 360 particles. Like with the ANNs, this is done every midnight and 24-h ARIMA predictions are made for each time series. The time series are allowed to grow continuously such that at the beginning of the simulation, the ARIMA models are crudely fit on only one days' worth of observations, while the models on the last day are fit on the full time series.

Let the error between predicted velocity (u^p, v^p) and the real target values (u^r, v^r) for all i prediction times (06, 12, 18, 24 h), j test particles, k forecasts issued throughout the simulation, and l ANN ensemble members per experiment be defined as:

$$e_{ijkl} = \left[\left(u^r_{ijkl} - u^p_{ijkl} \right)^2 + \left(v^r_{ijkl} - v^p_{ijkl} \right)^2 \right]^{1/2} \tag{6}$$

Averaging over j, k, and l quantifies the 24-h forecast errors for the entire experiment:

$$\overline{e_i} = \frac{1}{pfm} \sum_{j=1}^{p} \sum_{k=1}^{f} \sum_{l=1}^{m} e_{jkl} \tag{7}$$

where $p = 90$ particles, $f = 13$ for Cases 1 and 3 and $f = 28$ for Case 2, and $m = 3$ ensemble members. Average forecast errors are calculated between the ANN predictions and those from both persistence and ARIMA models.

3.2. Results: Proof-of-concept

Predicted trajectories are generated by splitting successive predicted velocities into 30 min increments using linear interpolation in order to minimize RK4 time step integration errors. ANN performance for Cases 1, 2, and 3 are summarized in Figs. 4, 5, and 6, respectively. A representative test particle trajectory during the last two days of the simulation is shown in panel (a) by a solid gray line with forecast times marked by circles. The dotted blue and dashed green lines illustrate the trajectories predicted by the one-to-one and time series ANNs, respectively. ARIMA-predicted trajectories are indicated by orange dashed–dotted lines and persistence predictions by solid red for comparison. Forecast errors from Eqs. (6)–(7) for velocity and position are shown in panels (b) and (c), respectively, with error bars indicating the spread of the forecasts given by one standard deviation from the mean error.

The simple rotational flow of Case 1 and the spiral trajectories of Case 2 proved to be trivial scenarios for both the one-to-one and the time series ANNs to learn. Predicted trajectories from both networks are visually indistinguishable from each other and also from the real trajectory itself (Figs. 4 and 5a). This offered stark improvement over the persistence forecasts which, as expected for circular trajectories, are always tangent to the eddy. Both networks for Case 1 had similar velocity errors of approximately $0.7 \pm 0.4\,\text{cm s}^{-1}$ and position errors around $50 \pm 50\,\text{m}$ at hour 6, increasing to around $450 \pm 300\,\text{m}$ at hour 24. Case 2 velocity error increased from about $0.3 \pm 0.3\,\text{cm s}^{-1}$ at hour 6 to about $0.7 \pm 0.5\,\text{cm s}^{-1}$ at hour 24 for the one-to-one ANN, which corresponded to position errors around $32 \pm 30\,\text{m}$ at hour 6 increasing to

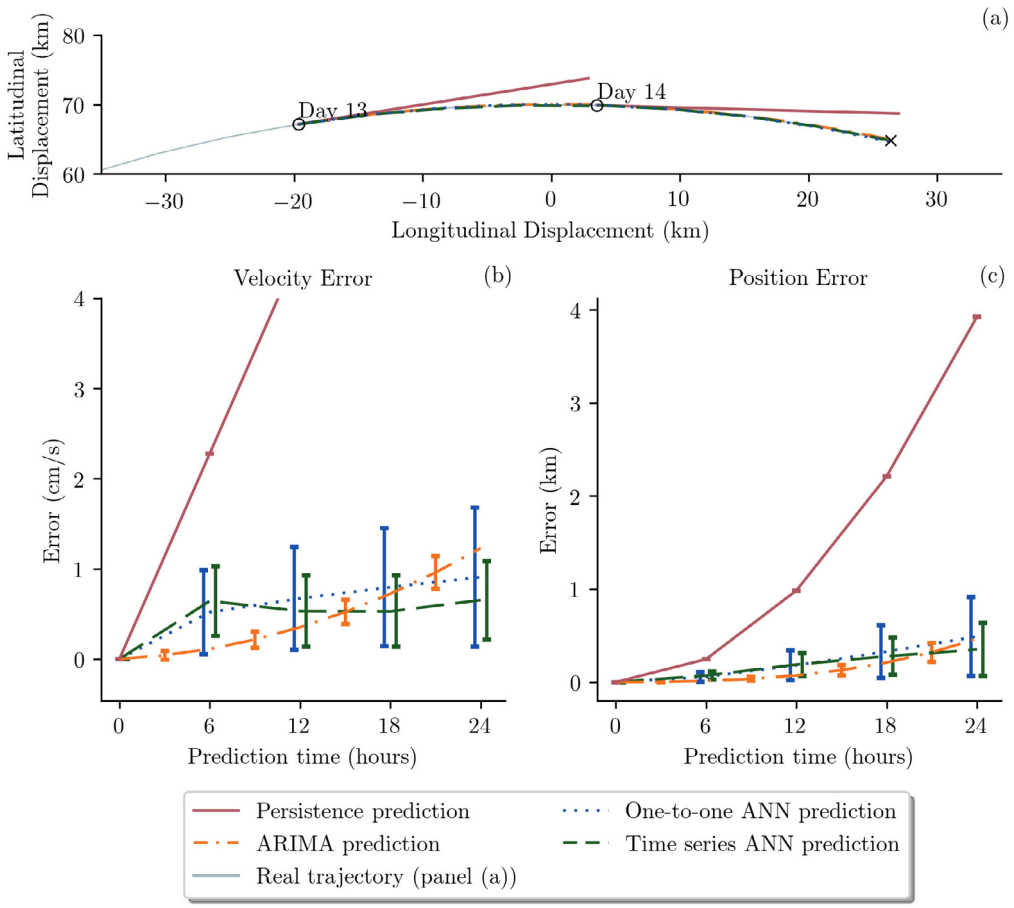

Fig. 4. Forecast error for proof-of-concept case 1 (stationary eddy). (a) Particle trajectory for one test particle (gray solid line) against predicted trajectories from one-to-one ANN (blue dots), time series ANN (green dashes), ARIMA (orange dash-dots), and persistence (solid red) models. Forecasts are for days 13 and 14, issued at 00:00 each day (circles). (b) Mean velocity error for persistence (solid red line), one-to-one ANN (blue dotted lines), time series ANN (green dashed line), and ARIMA models (orange dash-dots) vs. prediction time. Errors are averaged over all daily forecasts, particles, and ANN ensemble members (see text), with error bars denoting one standard deviation from the mean. (c) Same as (b) but for particle positions derived from the respective velocities.

around 740 ± 550 m at hour 24. The time series ANN velocity errors were fairly constant around $0.4 \pm 0.3\,\mathrm{cm\,s^{-1}}$ throughout the entire prediction window, leading to position errors increasing from around 47 ± 38 m at hour 6 to around 224 ± 216 m at hour 24.

The ARIMA ensemble exhibited a smaller standard deviation than the ANNs, especially in Case 1, but the average error increased almost linearly throughout the prediction window. While the one-to-one ANN behaved similarly, the time series ANN mean error remained fairly constant in both cases. The slight underperformance by the one-to-one ANN relative to the time series ANN beyond 12 h can be traced to the first forecast issued 00:00 on Day 2. This early version of the one-to-one ANN produced velocity prediction errors at hour 24 of nearly $7\,\mathrm{cm\,s^{-1}}$ for several particles in Case 1 and $4\,\mathrm{cm\,s^{-1}}$ in Case 2. These outliers aside, both ANNs performed equally well in both of these cases.

Case 3 is more interesting. This time ARIMA and persistence were comparable and the one-to-one ANN performed just as poorly as both, while the time series ANN offered notable predictive improvement. The trajectory in Fig. 6a illustrates a loop on Day 13 that scales as a submesoscale feature. A 180° turn like this might be observed in nature if a change in weather pattern, such as the passage of a strong atmospheric front, reverses the direction of the surface flow over the course of several hours. The one-to-one ANN predicted a curved trajectory (blue dots), but fell far short of anticipating the sharp curve that the particle undertook. The ARIMA model predicted a similar curved trajectory as the one-to-one ANN, but the velocities were nearly twice those of the ANN. This took the particle twice as far away. Consequently, by the end of the 24-h period, the particle ended up being about 55 km away from its predicted position. This was only mildly better than the persistence prediction, which put the particle roughly 75 km away from its actual position. In comparison, the time series ANN (green dashes) successfully predicted the particle's change in direction on Days 13 and 14, with the final location being about 3 km away from the prediction.

Error analysis revealed that the mean velocity error at hour 24 for the time series ANN was about $10 \pm 9\,\mathrm{cm\,s^{-1}}$ compared to $>52 \pm 44\,\mathrm{cm\,s^{-1}}$ for the one-to-one ANN, which led to a 5-fold decrease in position error. These results are corroborated by other particles with a variety of trajectory shapes as demonstrated in Fig. 7. Most trajectories contain sections that are relatively easy to predict – for example, the last few days in Fig. 7d – and other sections that are more challenging, such as the loops in the northern and southern regions of the domain. Overall, passing a time series of observations to the network as input significantly improves the ANN's predictive ability, highlighting the importance of network architecture and problem setup.

4. Application: Gulf of Mexico mesoscale eddy

4.1. Methods

To test the ANNs in realistic ocean flows, we set up a case study using output from a high-resolution 1/12° HYbrid Coordinate Ocean Model (HYCOM) for a mesoscale eddy observed in the Gulf of Mexico during January 2010 (see Prasad and Hogan, 2007 and Haza et al., 2016 for full details of model configuration). This anticyclonic mesoscale circulation surrounded by submesoscale structures provides

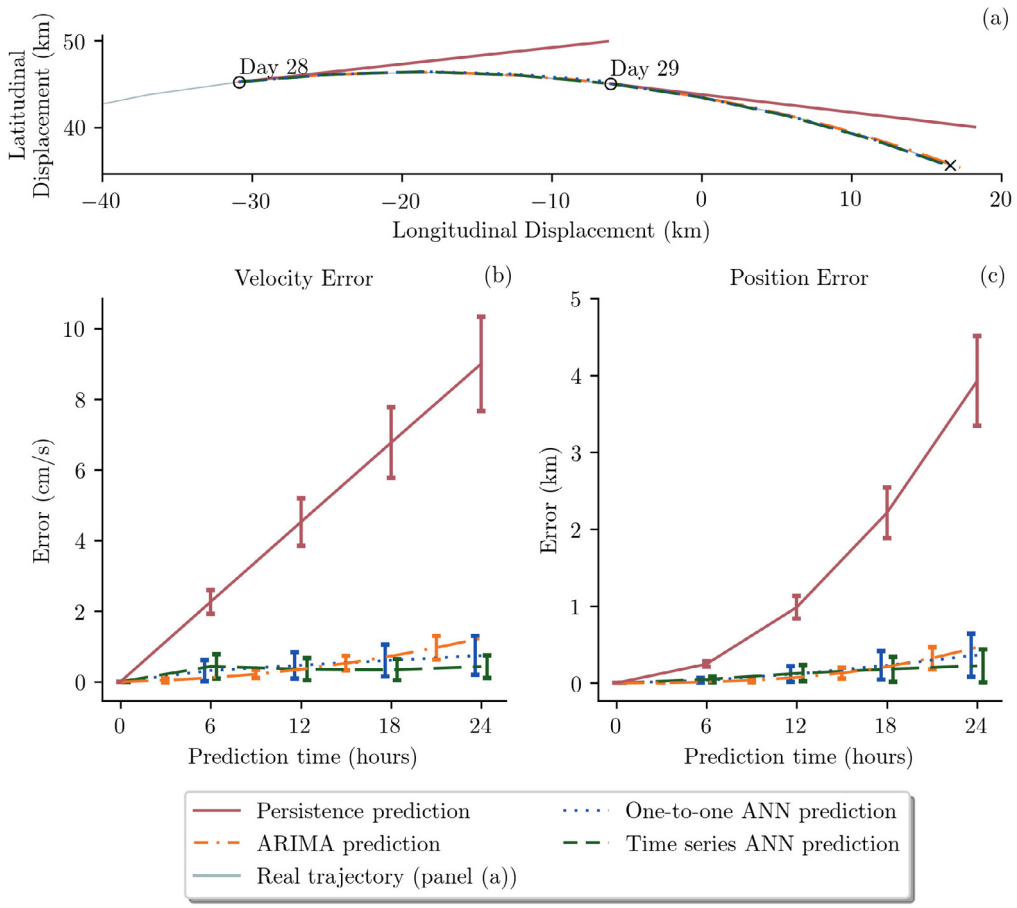

Fig. 5. Forecast error for proof-of-concept case 2 (moving eddy). (a) Particle trajectory for one test particle (gray solid line) against predicted trajectories from one-to-one ANN (blue dots), time series ANN (green dashes), ARIMA (orange dash-dots), and persistence (solid red) models. Forecasts are for days 28 and 29, issued at 00:00 each day (circles). (b) Mean velocity error for persistence (solid red line), one-to-one ANN (blue dotted lines), time series ANN (green dashed line), and ARIMA models (orange dash-dots) vs. prediction time. Errors are averaged over all daily forecasts, particles, and ANN ensemble members (see text), with error bars denoting one standard deviation from the mean. (c) Same as (b) but for particle positions derived from the respective velocities.

a known example of interacting scales of motion and a realistic manifestation of our proof-of-concept Case 3.

Particles are advected through the eddy in the same manner as before: 360 equally-spaced particles released simultaneously at 00h on 15 January 2010 in a circle of radius 70×10^3 m, a rough estimate of the perimeter of the mesoscale circulation. Particles are advected for 14 days, ending on 28 January. HYCOM Eulerian velocity fields were generated every 3 h; thus, spatial and temporal interpolations are necessary when implementing the RK4 integrator to produce trajectories from the model output. Acceleration fields are calculated between successive velocity output in order to interpolate velocity on time steps smaller than 3 h. Spatial interpolation for both velocity and acceleration is carried out using cubic 2D interpolation. Data setup, network configurations, and training methods are as described in Section 3.1

The time series ANN results of Case 3 raise questions regarding sensitivity to the chosen prediction window. If the forecast is extended beyond 24 h, will the time series ANN perform just as well, or will its error climb to those of the other models? We investigate this in the context of this realistic eddy by considering 72-h predictions at each forecast time without changing the configuration of our ANNs. The output time series is interpolated to 3-h time steps using linear interpolation and passed back to the network twice, as described in Section 3.1.4.

Acknowledging the practical challenges of deploying hundreds of drifters at sea, we also consider the sensitivity to the number of test particles by repeating this 72-h forecast simulation releasing a total of 90 and 45 particles. Here we utilize a 5-fold cross-validation technique to assess the skill of the networks in each scenario. k-fold validation is easily implemented by randomly dividing all particles into $k = 5$ groups without replacement. Each of these groups is then taken to be unique set of test particles and the remaining particles become the training examples for the given k-fold. This allows every particle to be predicted once by an ANN. Instead of initializing three unique random ANNs as above, we seed the random number generator so that every ANN starts in the same solution space and its evolution depends only on the particles in the respective training set.

Finally, we explore different numbers of hidden layers (1, 2 and 3) and hidden neurons (10, 20, 50, 100 for one-to-one; 20, 50, 100, 200 for time series ANNs) using 360 particles. Three uniquely initialized models are trained for each test, as before, to account for variations in learning due to random initialization. We hereafter refer to the HYCOM velocity output as observations in order to differentiate them from the ANN-predicted velocities.

4.2. Results: Gulf of Mexico mesoscale eddy application

The Gulf of Mexico mesoscale eddy provides a far more realistic representation of what was idealized in Case 3 above. One important difference is that the particles in this scenario are no longer trapped within the mesoscale circulation. Instead, some escape – mostly from the northern rim of the eddy – while some end up being drawn into the center of the eddy and still others stay around the perimeter of the mesoscale circulation (Fig. 8). A second difference is that the fluctuating velocity component is no longer sinusoidal — or, at the

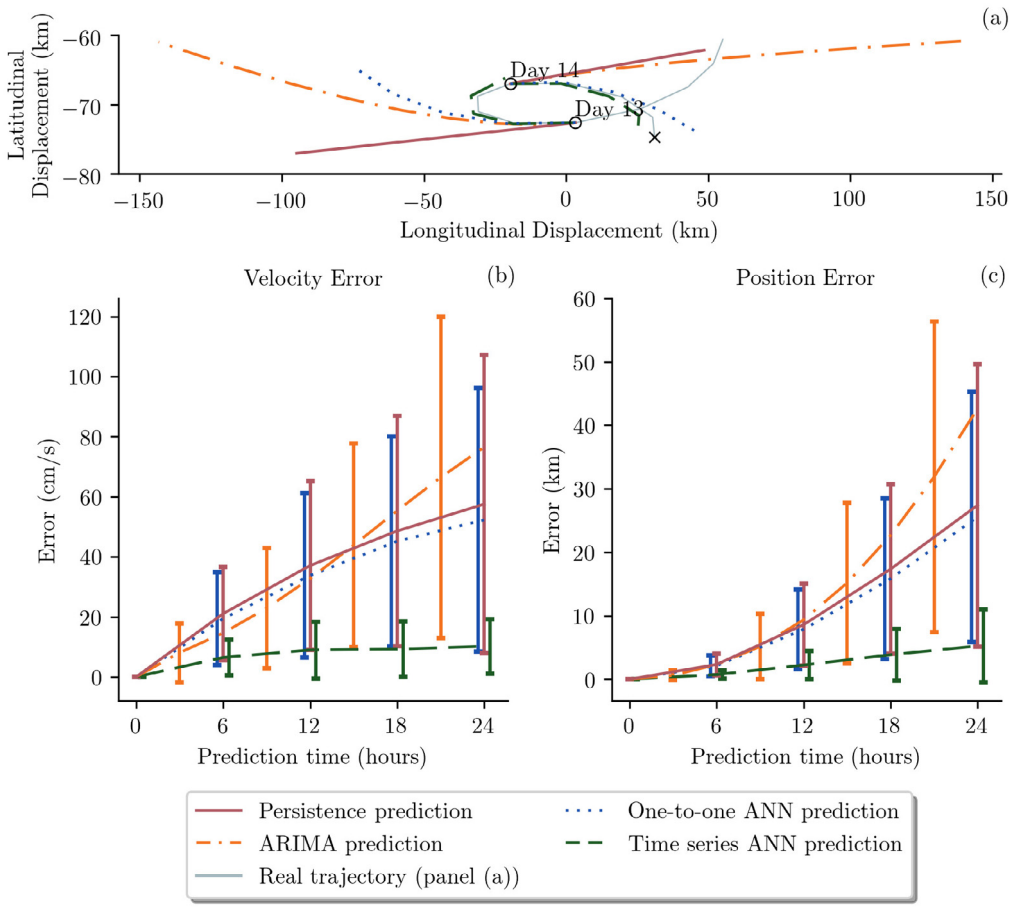

Fig. 6. Forecast error for proof-of-concept case 3 (moving eddy with noise). (a) Particle trajectory for one test particle (gray solid line) against predicted trajectories from one-to-one ANN (blue dots), time series ANN (green dashes), ARIMA (orange dash-dots), and persistence (solid red) models. Forecasts are for days 13 and 14, issued at 00:00 each day (circles). (b) Mean velocity error for persistence (solid red line), one-to-one ANN (blue dotted lines), time series ANN (green dashed line), and ARIMA models (orange dash-dots) vs. prediction time. Errors are averaged over all daily forecasts, particles, and ANN ensemble members (see text), with error bars denoting one standard deviation from the mean. (c) Same as (b) but for particle positions derived from the respective velocities.

very least, there are enough interacting dynamics that any predictable periodicity is much harder to discern. This enables us to test whether or not the time series ANNs in Case 3 were actually learning the superimposed periodic signals. If they were, then they should struggle to learn these realistic trajectories.

Fig. 9 summarizes the error results for the one-to-one and time series ANNs trained on these eddy trajectories. The ARIMA errors (both mean and standard deviation) are indistinguishable from persistence, and like with the proof-of-concept Case 3, the one-to-one ANN does not offer any improvement over either metric. The velocity and position error plots indicate that, on average, the one-to-one ANN predictions at hour 24 were worse than both of these models. The representative test particle shown in Fig. 9a illustrates how this might happen. The one-to-one ANN predicted for Day 13 that the particle would continue its slight clockwise curve that it had been tracing the day before. Instead, the particle curved back to the left before moving due west. As a result, its final position ended up closer to the persistence prediction than the one-to-one prediction. While this seems fluky, a similar situation played out on Day 14. The time series ANN, on the other hand, better predicted the slightly wavy trajectory. This may be because the particle's trajectory during Day 12 was also somewhat *s*-shaped. The time series ANN velocity errors leveled off around $7.2 \pm 4.5\,\mathrm{cm\,s^{-1}}$, and the position errors climbed from around $1\,\mathrm{km}$ at hour 6 to around $5 \pm 3.3\,\mathrm{km}$ at hour 24. This also illustrates the error cascade from velocity to position: even if velocity errors stop growing, they will still propagate through to positions and affect the entire predicted trajectory. ARIMA forecasts for Days 13 and 14 were not very different from those of the time series ANN. Interpretation of the ARIMA results will be discussed further in the next section.

Time series ANN performance on longer range 72-h forecasts is shown in Fig. 10 for a total of 360, 90, and 45 particles. With 5-fold validation, these networks were trained with 288, 72, and 36 particles, respectively. The dashed gray lines indicate the forecast iteration times when the model output were used to generate the next 24-h prediction. ARIMA models were fit as before using the *auto.arima* routine, but full 72-h predictions were made at each forecast time. All model errors were calculated using Eqs. (6)–(7). Fig. 10 shows mean error with one standard deviation.

Several notable take-aways follow from Fig. 10. First, after the initial 24 h, ANN mean velocity error steadily increases towards – though never surpasses – the mean ARIMA model error, roughly doubling from almost $10\,\mathrm{cm\,s^{-1}}$ at hour 24 to approximately $23\,\mathrm{cm\,s^{-1}}$ at hour 72. Secondly, while ANN position error increases 5-fold from approximately $5\,\mathrm{km}$ at hour 24 to approximately $25\,\mathrm{km}$ at hour 72, it remains well below that of ARIMA through hour 72 such that even the upper bounds of the ANN error envelope are at or only slightly greater than the ARIMA mean position error. This is promising because position is often of greater interest to the particle transport problem than velocity. Third, the ANN standard deviations remain smaller throughout the prediction window. Finally, for the first half of the prediction window (0-36 h), the mean error for the ANNs with 360 particles is slightly lower than the ANN with 90 particle and the error spread somewhat smaller, as should be expected (and similarly for 90 particles versus 45). After hour 36, there is little statistical difference between the three

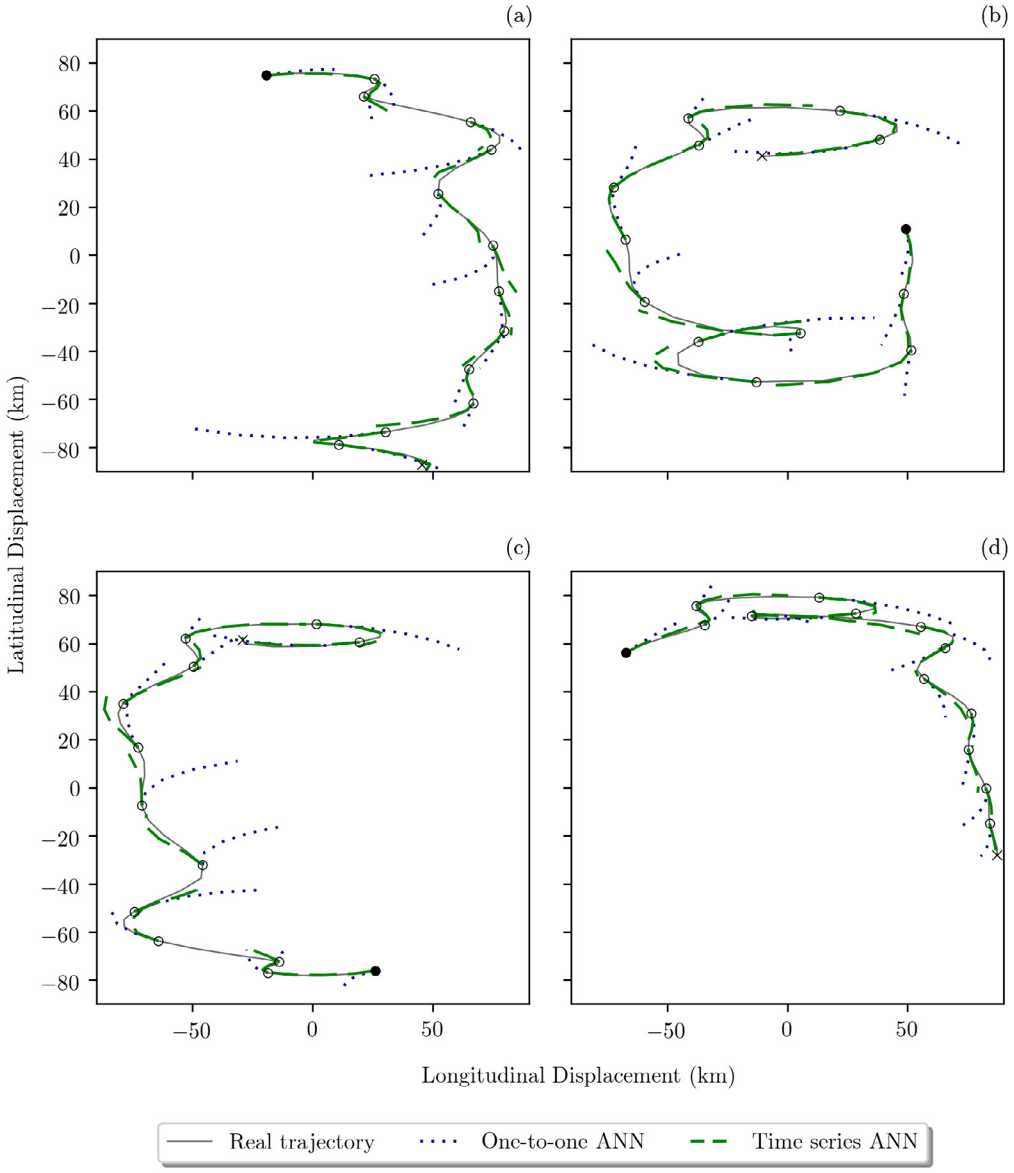

Fig. 7. Representative particle trajectories (solid line) for proof-of-concept case 3 with predicted trajectories from one-to-one ANN (blue dotted lines) and time series ANN (green dashed lines).

configurations, though the 360-particle ANN has a smaller position forecast spread throughout the entire window. We discuss each of these outcomes in the next section.

We next investigate whether changing the size (number of neurons) and depth (number of layers) of both one-to-one and time series ANNs helps the networks learn to predict trajectories in realistic flows. Let the convention $m \times n$ describe these networks by the number of hidden layers, m, and the number of hidden neurons in each layer, n. To assess the performance of each network configuration, we calculate the relative error between the real target values, x^r, and the predicted values, x^p:

$$\mathrm{RE}_{ijkl} = \left| \frac{x^r_{ijkl} - x^p_{ijkl}}{x^r_{ijkl}} \right| \tag{8}$$

where x^r is either the ANN output or the persistence prediction and the subscripts are as defined in Section 3.2. Fig. 11 shows relative error ratio between positions derived from persistence predictions and those derived from ANN predictions averaged over i, j, and k:

$$\overline{\mathfrak{R}_l} = \frac{1}{gpf} \sum_{i=1}^{g} \sum_{j=1}^{p} \sum_{k=1}^{f} \frac{|x^{\mathrm{real}}_{ijk} - x^{\mathrm{pers}}_{ijk}|}{|x^{\mathrm{real}}_{ijk} - x^{\mathrm{pred}}_{ijk}|} \tag{9}$$

where $g = 4$ predictions per forecast, $p = 90$ particles, $f = 13$ forecasts for the eddy simulation. Error bars indicate standard error. All ratios are >1 indicating that every ANN outperformed persistence. Despite larger networks taking five to six times longer to train, no appreciable difference in performance is observed as the ANNs increase in complexity. One explanation for this is that the hyperparameters – such as learning rate and number of training epochs – were not tuned for each model but rather were kept constant in order to assess the affect of changing either m or n on model performance.

Finally, we consider whether or not the neural networks continued to learn throughout the simulations due to the continuous training method. Fig. 12 shows position error from Eq. (6) averaged over i prediction times and j test particles for the one-to-one ANNs (blue dots), time series ANN (green dashes), and ARIMA model (red dash-dots). The ARIMA model exhibits a clear downward trend from an average position error of almost 9 km at the start of the simulation around 2 km on the last day. This can be attributed to the fact that early in the simulation, the time series were too short to successfully regress. In these scenarios, the ARIMA forecasts were nothing more than the time-mean of the few available observations. As the time series grew,

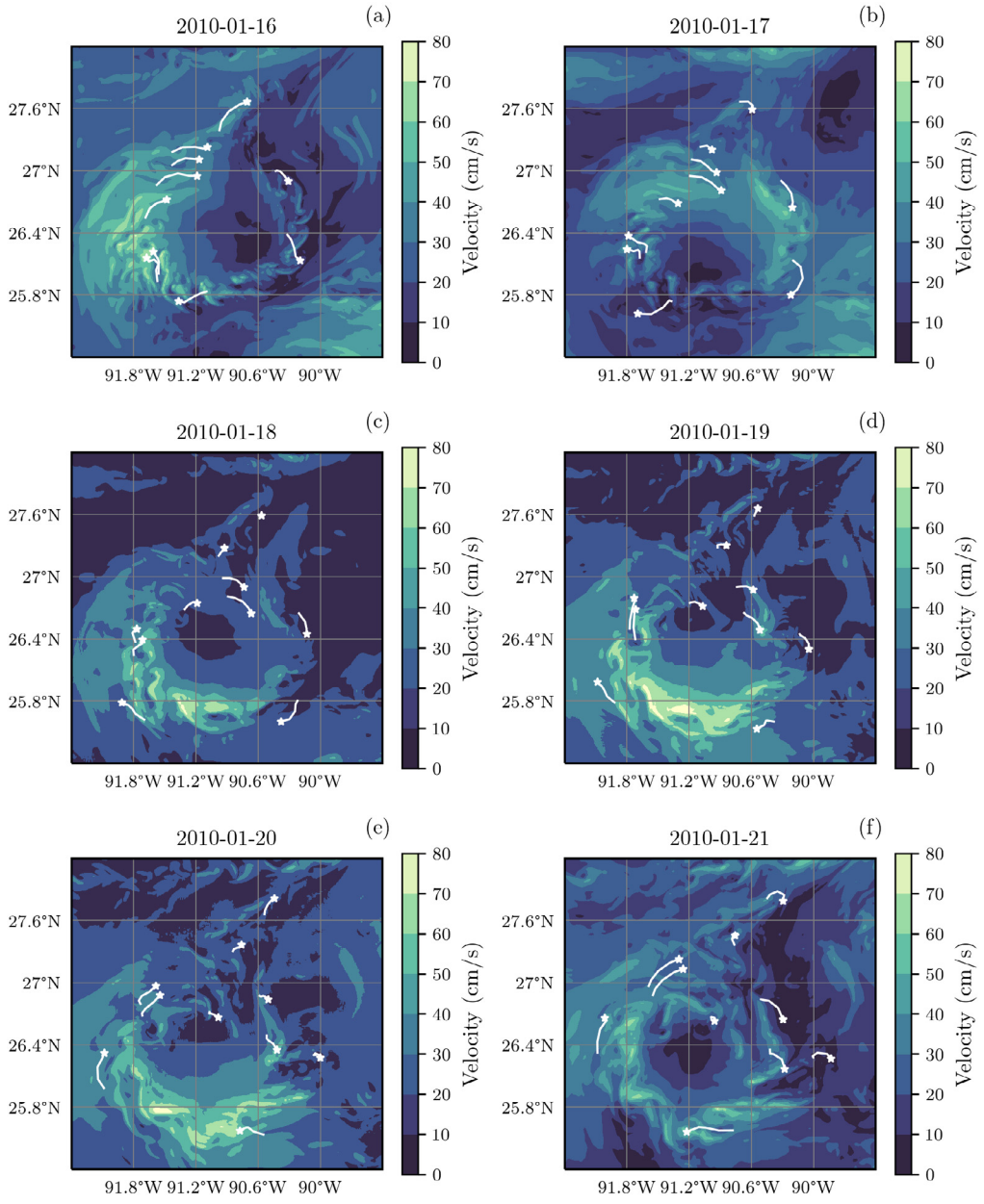

Fig. 8. Sample trajectories of particles in a HYCOM flow field for a Gulf of Mexico anticyclonic mesoscale eddy, 16–27 January 2010.

this became less of a problem and the ARIMA models acquired greater skill. In contrast, no significant trends are observed for any of the neural networks, though the one-to-one ANNs seem to improve from around 9 km on 16 Jan to around 5 km by 28 Jan. Overall, we conclude that while the majority of the learning by both network configurations took place during the initial training session, subsequent training routines kept the networks up-to-date using the latest information. For time-varying domains, this is equally as important as initial learning.

5. Discussion

We demonstrated preliminary success at predicting particle transport in the ocean using simple artificial neural networks. Two questions motivated this study. First, inspired by the ability of human scientists to discern patterns in oceanic flows (*e.g.*, Rupolo, 2007), we asked: can a data-driven machine learning technique accomplish a similar task? The goal was to extract enough information from ground truth observations to make respectable forecasts instead of starting with primitive equations. The second question was: can a data-driven algorithm learn to make accurate ocean forecasts *without knowing anything whatsoever about the underlying physics?* If so, these tools could potentially supplement existing theory-driven OGCMs.

The proof-of-concept experiment provided a starting point for testing idealized ocean trajectory patterns, such as simple inertial oscillations (*e.g.*, Beron-Vera et al., 2015; Gough et al., 2016) and interacting scales of motion (*e.g.*, Haza et al., 2016). The most promising results were the abilities of these simple ANNs to predict trajectories within and around a realistic Gulf of Mexico mesoscale eddy. We attribute this success at least partially to the continuous-learning training scheme whereby training continues every time the oldest observation is replaced with a new one. While the more common approach is to train a neural network once using a large data set and then use it to make predictions of new data, this would require having in advance an impractically large training set consisting of full trajectories for millions of particles. Instead, we treated the domain as continuously varying in time and allowed the network to keep up with the latest available observations.

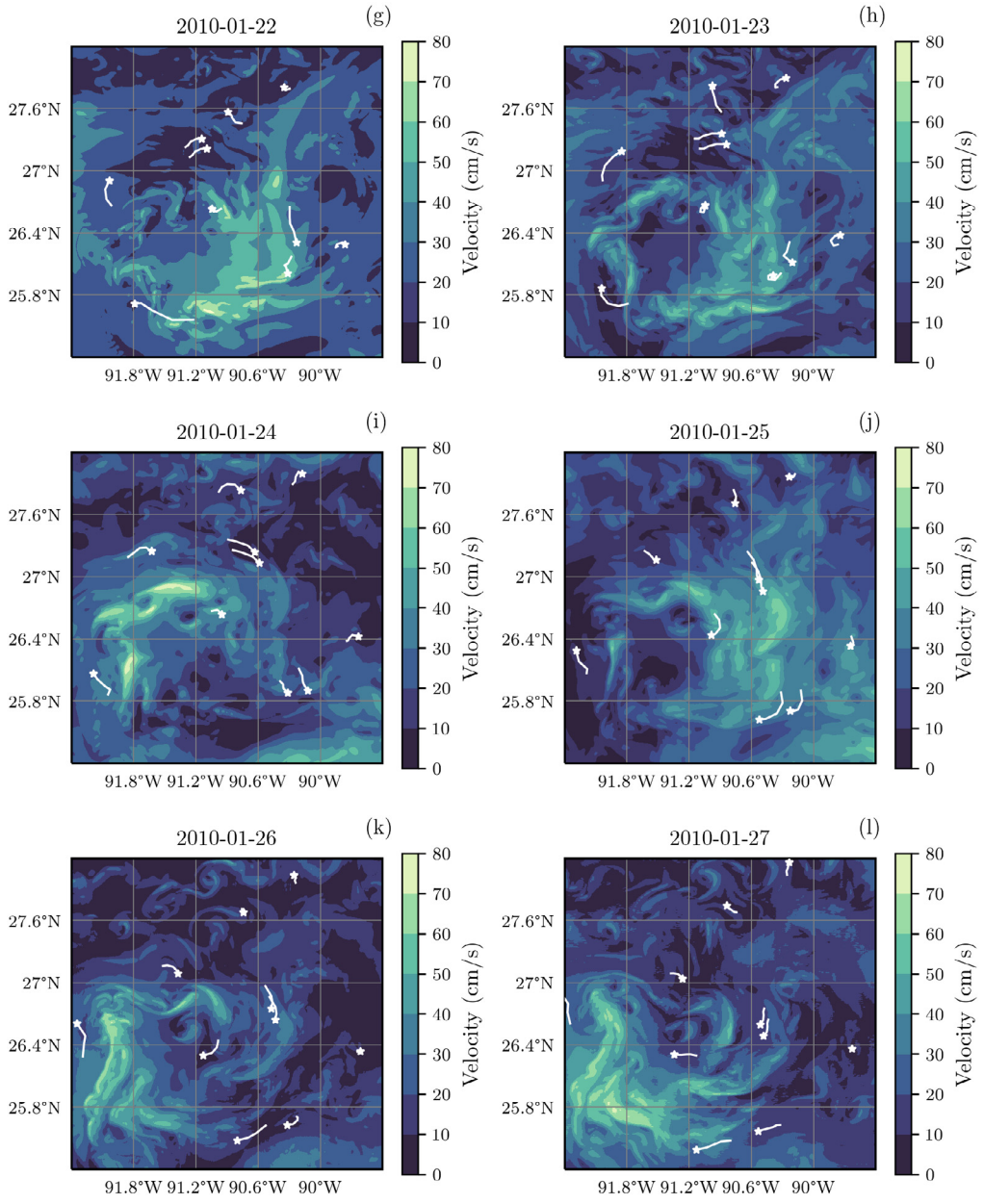

Fig. 8. (*continued*).

The contrasting performance of one-to-one and time series ANNs in realistic flows demonstrated the importance of using theory to guide model development (*e.g.*, Faghmous et al., 2014). The one-to-one ANNs have the operational advantage of only requiring one velocity observation to make a forecast, but trained this way, they lack any concept of time and can only "remember" the most recently seen example. Changing the input to a time series of observations significantly improved performance, since the network was able to consider where the particle had been over some period of time. This enhanced performance is to be expected, given the fundamental nature of time series, and is also in agreement with the first notable discussions on handling time and memory in learning paradigms by Jordan (1986) and Elman (1990, 1991).

It is reasonable to ask how the time series neural network managed to outperform even ARIMA forecasts over a 24-h prediction window. The primary reason for this is that ARIMA models are inherently sensitive to time series length. The wide ARIMA forecast spread (large standard deviations in Figs. 6, 9, and 10) is due to the time series early in the simulation being too short to be properly regressed (*e.g.*, 16–18 Jan in Fig. 12). In these cases, *auto.arima* returned models whose forecasts were merely the time-mean of the few available observations. At least 72 h of observations were necessary before skill became comparable to the one-to-one network, where it remained for most of the simulation (18–26 Jan). Not until the last three days of the simulation did the ARIMA model have a sufficient number of data points to make predictions similar to those of the time series ANN.

While the ARIMA models could only be fit to individual time series, the ANNs had the advantage of being able to learn from an assortment of unique time series. Thus, both the one-to-one and time series ANNs demonstrated greater skill than ARIMA during the first two days of the simulation (Fig. 12). It is worth noting that the skill of both ANN configurations came from seeing only 24-h worth of observations at any given time, and the majority of the networks' learning occurred during the initial training session (that is, the training leading up to the 06 Jan forecast shown in Fig. 12). ANNs may therefore be favorable for generating time-sensitive forecasts soon after deploying observational drifters, whereas one may need to wait several days before a traditional regression model could provide any skilled insight.

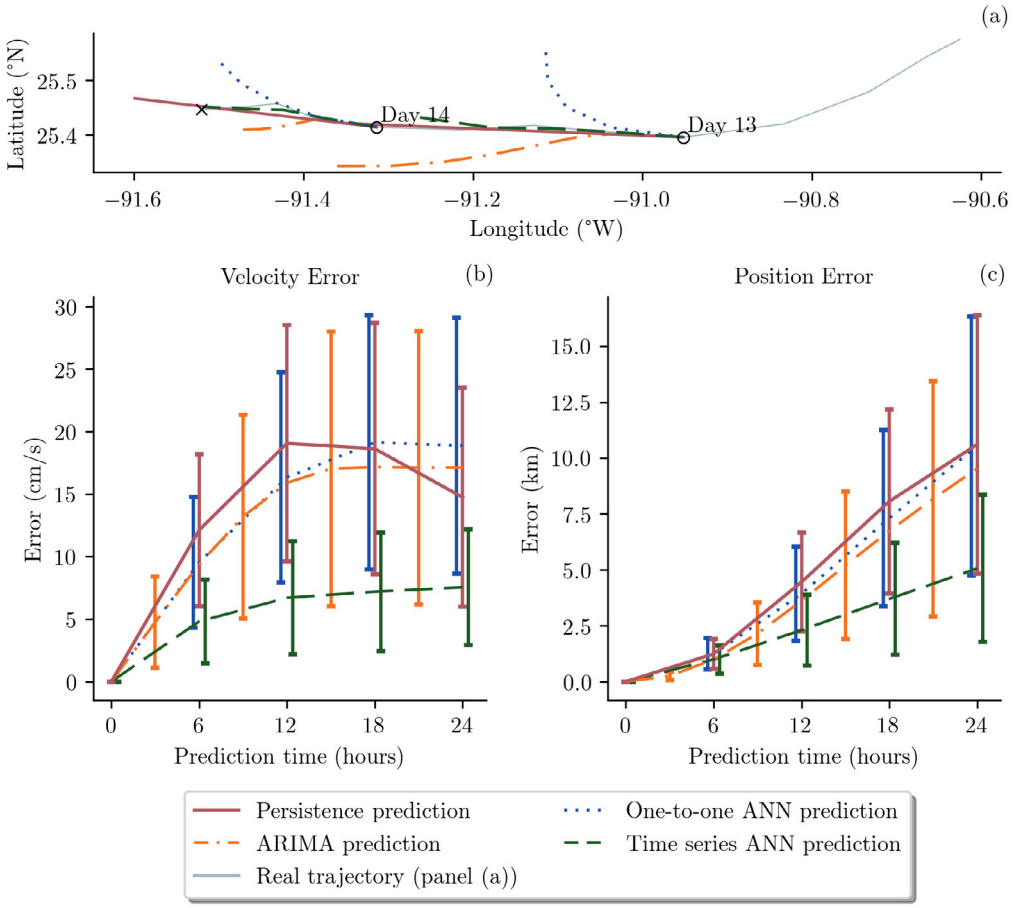

Fig. 9. Forecast error for Gulf of Mexico eddy application test. (a) Particle trajectory for one test particle (gray solid line) against predicted trajectories from one-to-one ANN (blue dots), time series ANN (green dashes), ARIMA (orange dash-dots), and persistence (solid red) models. Forecasts are for days 13 and 14, issued at 00:00 each day (circles). (b) Mean velocity error for persistence (solid red line), one-to-one ANN (blue dotted lines), time series ANN (green dashed line), and ARIMA models (orange dash-dots) vs. prediction time. Errors are averaged over all daily forecasts, particles, and ANN ensemble members (see text), with error bars denoting one standard deviation from the mean. (c) Same as (b) but for particle positions derived from the respective velocities.

It is well known that model accuracy inevitably breaks down when a learning machine is used to make predictions outside of the domain in which it was trained. This is precisely why the predictive capability of our ANN erodes beyond 24 h (Fig. 10). This shortcoming might be overcome by modifying our ANN to instead predict the full 72-h time window, but doing so would require target data at times $(t_o + 6\,\mathrm{h}, t_o + 12\,\mathrm{h}, \ldots, t_o + 72\,\mathrm{h})$ for each training example in addition to input observations from the previous 24 h. Thus, such a configuration would require 96 h of training data.

A final caveat of our experimental setup worthy of consideration is that although the ANNs are evaluated on particles not seen during training, predictions are made over the same time period as the training target data. This is fundamentally different from time series regression models that do not require future data. While both ARIMA and ANNs are function approximation models, the neural network represents a single mapping function developed through exposure to many representative time series in order to describe other similar time series. This also explains the relatively small sensitivity to the number of training particles: the unseen particles are most likely to be predicted relative to proximate training particles. Thus, our results suggest that one can use only $\mathcal{O}(100)$ drifters to forecast where intermediate drifters would go, as long as the drifters all sample the same dynamic features. This is especially advantageous for the oceanographer since deploying vast arrays of drifters is expensive and often infeasible.

We conclude by commenting on overfitting, an inherent risk that plagues all machine learning paradigms wherein a model learns the training data too well and then fails to extrapolate to new data. The ANNs in proof-of-concept Cases 1 and 2, for example, may have been prone to overfitting, as evidenced by velocity prediction errors greater than what might be expected for such trivial cases (Fig. 5 and especially Fig. 4). Aside from increasing the size of the training set, which is often not possible, overfitting can be minimized by decreasing the degrees of freedom of the model or minimizing the amount of training. The first strategy is easily accomplished by using the fewest possible hidden neurons. On the other hand, minimizing the amount of training is less trivial for systems that evolve continuously in time. Such systems often require continuous learning despite the increased risk of overfitting. We tested this by repeating the eddy simulation in the same way as described above except that the time series ANNs were not trained at all following the initial training on Day 1. This resulted in forecast errors that were consistently on par with those of both persistence and the worst ARIMA forecasts throughout the entire simulation (not shown), confirming that online learning was required. Ultimately, the selection of the best version of the ANN at each training session compromised between the need to keep the network up-to-date with the system dynamics while recognizing that, at any given time, little or even no additional training may actually be required.

To summarize, ANNs may be beneficial in real-ocean scenarios when:

(i) numerous time series are available that collectively contain useful information to assist with forecasting;
(ii) available time series are too short for regression models, *e.g.*, at the beginning of a coordinated drifter deployment; or

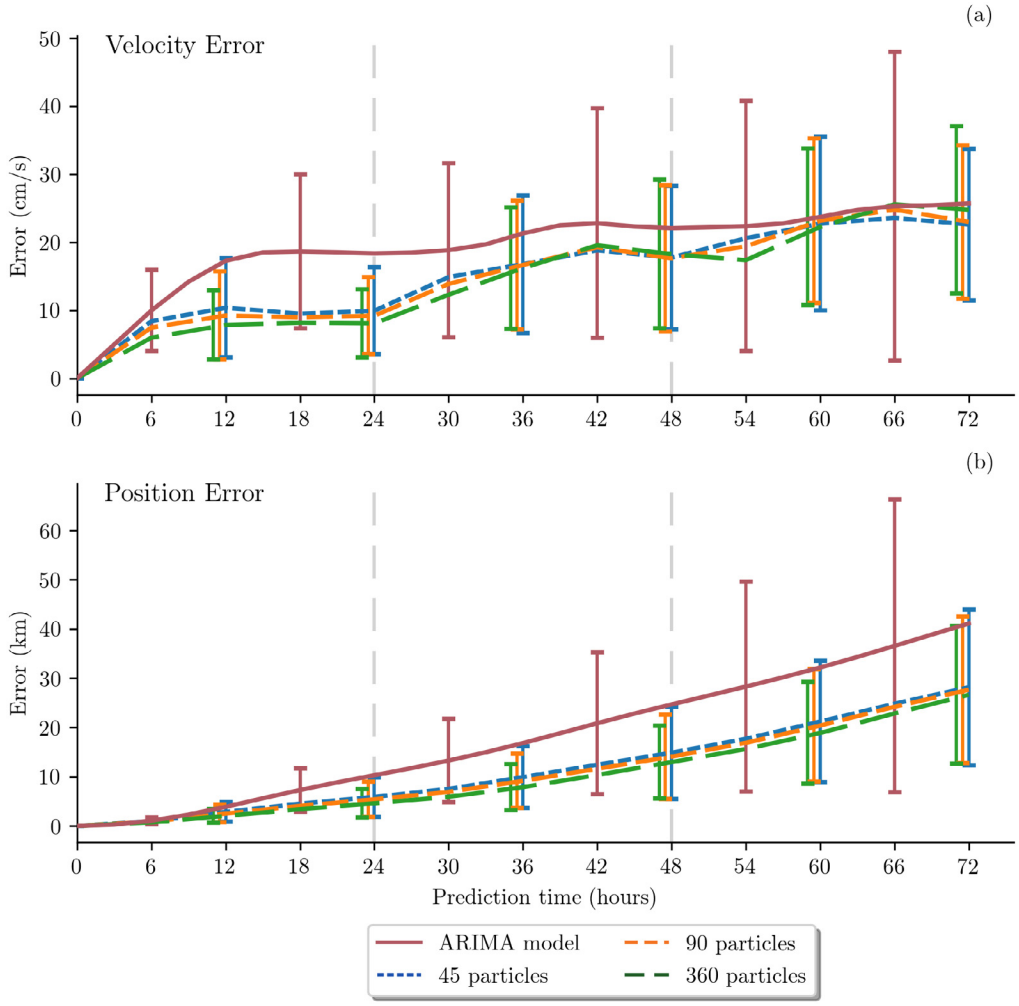

Fig. 10. Average extended 72-h forecast error issued by time series ANN for scenarios containing a total of 45 released particles (blue small dashes), 90 particles (orange medium dashes), and 360 particles (green large dashes). ANN forecasts are issued recursively such that the first output (0–24 h predictions) are fed back into the network to predict the next 24-h window; gray dashes mark these intervals. ANN performance is compared to 72-h ARIMA model forecasts. Errors are averaged over all daily forecasts and particles with error bars denoting one standard deviation from the mean.

(iii) predicting the transport of material within a local domain is necessary using observations dispersed within the domain, *e.g.*, predicting the spread of an oil plume using a limited collection of drifters.

6. Conclusions

We have demonstrated that particle trajectories in a variety of simulated and modeled oceanic flow regimes can be learned by simple artificial neural networks. The fundamental idea is to move away from relying on primitive equations of motion for prediction, as these are impossible to initialize and solve for all scales of motion in the ocean. Instead, the goal is to develop a predictive tool that learns from "ground truth" observational data without any direct physical guidance. We started with a proof-of-concept exploration using a hierarchy of idealized flows, and then tested our approach on realistic flows generated by a high-resolution Hybrid Coordinate Ocean Model for a mesoscale eddy in the northern Gulf of Mexico. This eddy provided a particularly illustrative example of interacting mesoscale and submesoscale dynamics and we showed that our approach to ANN training performed well even for these complex flow fields.

We tested two approaches to ANN training. In the first configuration, the network predicted a drifting particle's velocity at time $t_o + 6$ h using only its previous velocity at time t_o. This approach had the operational advantage of requiring only one velocity observation to make a forecast, but predictive performance suffered due to a lack of internal memory within the ANN. We also developed networks that predicted the drifting particle's velocity at time $t_o + i\Delta t, i \in (1, 2, 3, 4)$, based on the last 24 hours' worth of velocity measurements. This time series ANN performed between two and six times better than its one-to-one counterpart for both velocity and position estimates, depending on the complexity of the flow.

The predictive capabilities of the time series ANNs can be attributed to several things. First was allowing the ANN to "see" and learn from a time series of data. Second, we restricted our training data to the past 24 h of data, replacing old observations with the most recent. This helped the ANN forget regime shifts within the data set, such as a particle sharply changing direction or behavior, as may result from a strong weather event. We adopted a continuous training approach whereby ANN training continued whenever new data became available. This allowed the ANN to always be up-to-date with the constantly changing domain. Since real ocean drifters continuously provide new data, this training technique does not seem unreasonable. In fact, we argue that it presents a highly efficient and effective way of utilizing real-time data for ocean prediction. Finally, and most importantly, the time series ANN is able to learn from many independent trajectories, and these time series can be as short as 24 h (at 3-h increments). This is a fundamental advantage over traditional regression models that are

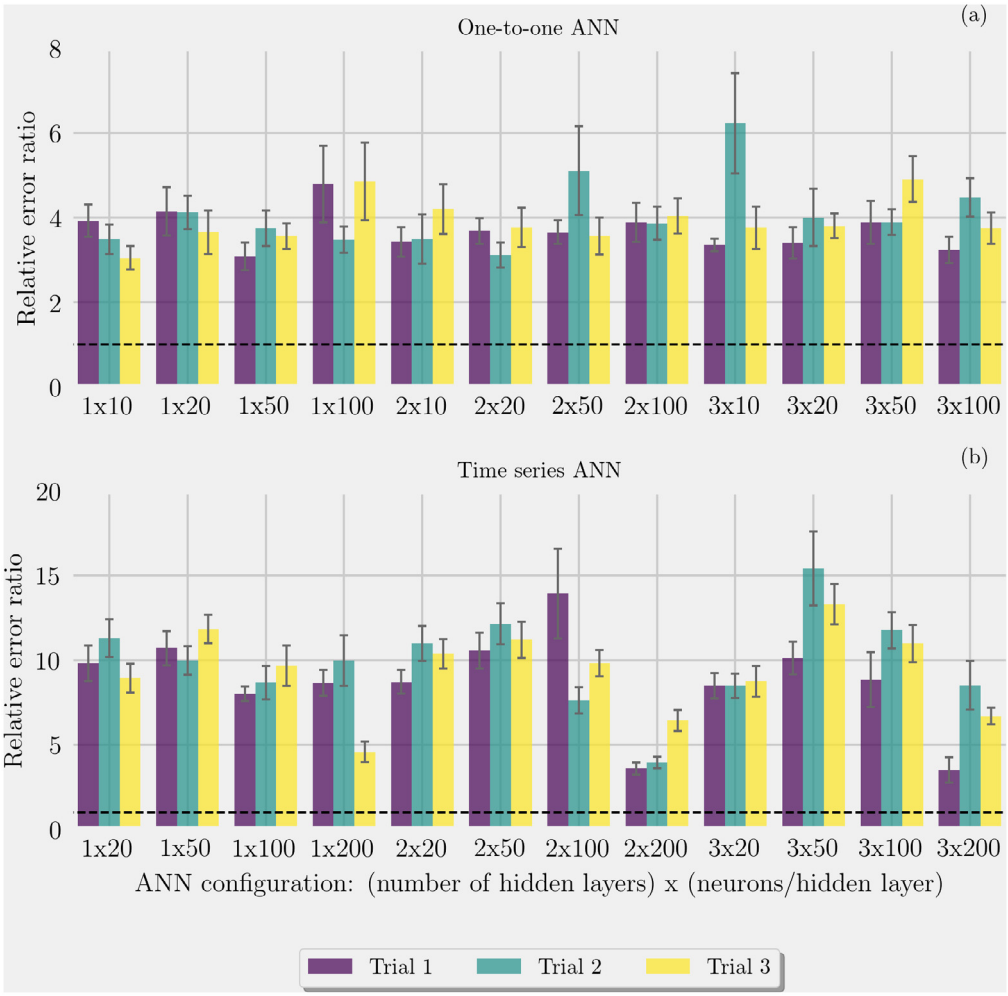

Fig. 11. (a) Relative error of persistence positions divided by relative error of ANN-derived positions for different one-to-one ANN configurations. (b) Same as (a) but for different time series ANN configurations.

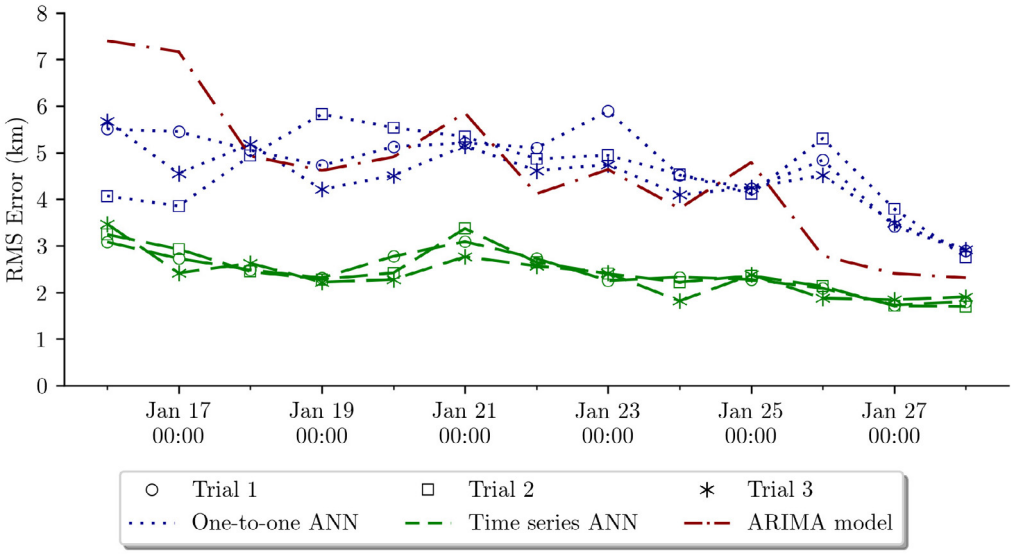

Fig. 12. Gulf of Mexico eddy test forecast errors for positions derived from one-to-one ANN (blue dotted lines), time series ANN (green dashed lines), and ARIMA model vs. time since particle release for three ANN ensemble members (symbols).

time-series-specific and, in the case of ocean particles, require at least 72 h of observations before any predictive skill is discernible.

This study addressed the question of whether a neural network could learn to predict a particle's velocity based only on its previous

velocities. An important distinction is in order between learning particle trajectories and learning the dynamics of the system in which the particles exist. The latter is a much more difficult problem because the sparsity of observational data means that the dynamics of the ocean are seldom known *a posteriori*. Learning to interpolate point-wise time series is often the best one can do when relying only on observations. From an operational standpoint, this should not be a concern: predicting the spread of an oil spill does not necessarily require learning or even knowing the full dynamics, but rather predicting how certain sections of a plume may evolve based on nearby observations.

The eddy application results suggest there is reason to be optimistic that ANNs will be able to learn observed trajectories just as well. Thus, future work will involve testing this predictive approach on real drifter data.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was made possible by a grant from the Gulf of Mexico Research Initiative, United States. Data are publicly available through the Gulf of Mexico Research Initiative Information & Data Cooperative (GRIIDC) at https://data.gulfresearchinitiative.org (doi:10.7266/N7BC3WJC). The authors thank the three anonymous reviewers whose constructive suggestions and recommendations greatly improved this manuscript.

Appendix. The artificial neural network

A.1. Brief overview

An artificial neural network (ANN) is a biologically-inspired nonlinear regression model that is trained to map input data to desired output values. Groups of nodes called neurons are arranged in layers such that adjacent layers are fully interconnected by weighted links, but no neurons within the same layer are connected to each other (see Figs. 2 and 3 in the text). Layers in between the input and output layers are commonly referred to as "hidden layers" because they are inside the network and the intermediate data mappings they produce are usually not known to the network engineer. All neurons consist of a trainable bias (or zero-th weight) that is added to the neuron's input signal and an activation function, usually chosen from families of functions that return values either on the interval [−1, 1] or [0, 1] (*e.g.*, Duch and Jankowski, 1999). Activation functions reduce the dimensionality of the data being passed through successive layers in the ANN.

All weights and biases are initialized as small random numbers. We take these from a univariate Gaussian distribution of mean 0 and variance 1. Training is the process by which solution space is searched for the optimum combination of these weights and biases such that the network output most accurately matches desired target values. A common approach is to pass training examples – a subset of all available data – to the ANN one at a time and compare the output to the corresponding target values for each example. If the output differs from the target, the error is back-propagated through the network to quantify each neuron's contribution to the overall error, and small adjustments are made to the weights and biases. The network is always tested and evaluated on data not seen during training.

We now describe this training process and the details about the ANN configuration used in this study. The discussion closely follows Kubat (2017). For a comprehensive and illustrative overview of neural networks, see also Nielsen (2014).

A.2. Network configuration and training

Consider a training example $\mathbf{x}$ defined as a vector containing n attributes describing that particular example: $\mathbf{x} = (x_1, x_2, \dots, x_n)$. Each attribute is assigned its own input neuron in the ANN. Thus, the one-to-one ANN described in Section 3.1.4 has $n = 2$ input neurons, where the attributes are horizontal velocity components. For the time series ANN, each example contains $n = 16$ attributes (8 observations × 2 velocity components; see Section 3.1.4 in the text), so these networks required 16 input neurons.

The simplest possible ANN contains a single hidden layer and as few hidden neurons as possible. We choose 10 hidden neurons for the one-to-one network and 20 hidden neurons for the time series ANN. Determining these sizes is more art than science, and we therefore conduct sensitivity tests in Section 4.

Let the weights connecting the kth attribute to the jth hidden neuron and the jth hidden neuron to the ith output neuron be denoted w_{kj} and w_{ji}, respectively. Similarly, let the biases for the k output neurons and the j hidden neurons be denoted b_k and b_j, respectively. A sigmoid transfer function of the form:

$$f(x) = \frac{1}{1 + e^{-\Sigma}} \tag{A.1}$$

is chosen for all hidden and output neurons, where $f(x)$ is the neuron output and Σ is the weighted sum from the previous layer in the network, *e.g.*,:

$$\Sigma_{j=1} = w_{11}x_1 + w_{21}x_2 + \cdots + w_{n1}x_n + b_1 \tag{A.2}$$

Each example is passed through the network as weighted sums of the attributes to produce an output vector $\mathbf{y} = (y_1, y_2, \dots, y_i)$:

$$y_i = f\left(\sum_j w_{ji} f\left(\sum_k w_{kj}x_k + b_j\right) + b_i\right) \tag{A.3}$$

In the case of the one-to-one ANNs, the output vector is the same length ($i = 2$) as the input vector. Thus, these networks also have $i = 2$ output neurons, one corresponding to zonal velocity and the other to meridional velocity, while the time series ANNs contained $i = 8$ output neurons (4 predictions × 2 velocity components).

Eq. (A.3) describes the behavior of a simple *feed forward neural network*. Note that since Eq. (A.1) returns values on the interval $[0, 1]$, the output of the ANN will also be on this interval. Thus, all data are normalized to this interval before being presented to the network, and all output must be returned to the range of the original data. This requires that the minimum and maximum values of the original data set be retained with the ANN and used to normalize all future data on which the trained ANN might be run.

To train the ANN, we first define a target vector $\mathbf{r} = (r_1, r_2, \dots, r_n)$ for each velocity observation. Here, $\mathbf{r}$ corresponds to a given particle's velocity at time $t_o + \Delta_p t$, where t_o is the issue time of the forecast and $\Delta_p t$ is the forecast prediction time step. Each training example is passed through the network (Eq. (A.3)) and the ANN output is compared to the target vector. Prediction error is then back-propagated through the network to quantify each neuron's responsibility to the overall error and the weights and biases are adjusted accordingly:

$$w_{ji} = w_{ji} + \eta\delta_i h_j, \qquad b_i = b_i + \eta\delta_i \tag{A.4a}$$

$$w_{kj} = w_{kj} + \eta\delta_j x_k, \qquad b_j = b_j + \eta\delta_j \tag{A.4b}$$

where $\eta \in (0, 1)$ is a small learning rate, h_j is the output of the jth hidden neuron, and δ_i, δ_j are quantifications of the neurons' error responsibility:

$$\delta_i = y_i(1 - y_i)(r_i - y_i) \tag{A.5a}$$

$$\delta_j = h_j(1 - h_j)\sum_i \delta_i w_{ji} \tag{A.5b}$$

Presenting the last training example to the ANN indicates the completion of one training epoch. The ANN performance is then evaluated

using a cost function whose value is to be minimized during the training process. The cost function in this study is taken to be mean squared error:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(r_i - y_i)^2 \tag{A.6}$$

Training continues for a defined number of epochs or until some performance threshold has been achieved. Once trained, the ANN is "run" by presenting new examples one at a time; predictions are made using Eq. (A.3) and the optimum weights and biases.

A.3. Final remarks

The strength of ANNs is their theoretical ability to fit any function. Adding neurons to an ANN introduces more weights to adjust and is comparable to introducing higher order terms to a nonlinear regression equation. Adding additional hidden layers increases the number of times the data are transformed within the model in order to reduce dimensionality as the network attempts to map the input vector to the desired output signal.

Like all regression models, however, neural networks are prone to overfitting the training data and thereby performing poorly on new data. This risk can often be minimized by decreasing the size of the network (*i.e.*, the number of hidden neurons), increasing the training data set, or by shortening the training time. Here, we adapt the smallest possible ANN and treat the completion of each training epoch as producing a new "version" of the ANN and take the best of these – that is, the one with smallest MSE – at the end of each 100-epoch training session. Another disadvantage of these models is that they often lack interpretability (see Section 5). Nevertheless, they can be ideal tools for problems in which one need not know why a particular outcome occurs. As discussed in the text, this is often true for predicting material transport in the ocean.

References

Aref, H., 1984. Stirring by chaotic advection. J. Fluid Mech. 143, 1–21.

Ayed, I., de Bézenac, E., Pajot, A., Brajard, J., Gallinari, P., 2019. Learning dynamical systems from partial observations. arXiv:1902.11136.

Barrick, D.E., Evans, M.W., Weber, B.L., 1977. Ocean surface currents mapped by radar. Science 198 (4313), 138–144.

Barrick, D., Headrick, J., Bogle, R., Crombie, D., 1974. Sea backscatter at HF: Interpretation and utilization of the echo. Proc. IEEE 62 (6), 673–680.

Berloff, P.S., McWilliams, J.C., 2003. Material transport in oceanic gyres. Part III: Randomized stochastic models. J. Phys. Oceanogr. 33 (7), 1416–1445.

Beron-Vera, F.J., LaCasce, J.H., 2016. Statistics of simulated and observed pair separations in the Gulf of Mexico. J. Phys. Oceanogr. 46 (7), 2183–2199.

Beron-Vera, F.J., Olascoaga, M.J., Haller, G., Farazmand, M., Triñanes, J., Wang, Y., 2015. Dissipative inertial transport patterns near coherent Lagrangian eddies in the ocean. Chaos Interdiscip. J. Nonlinear Sci. 25 (8), 087412.

Berta, M., Griffa, A., Özgökmen, T.M., Poje, A.C., 2016. Submesoscale evolution of surface drifter triads in the Gulf of Mexico. Geophys. Res. Lett. 43 (22), 11,751–11,759.

Bolton, T., Zanna, L., 2019. Applications of deep learning to ocean data inference and subgrid parameterization. J. Adv. Model. Earth Syst. 11 (1), 376–399.

Brunton, S.L., Proctor, J.L., Kutz, J.N., 2016. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proc. Natl. Acad. Sci. 113 (15), 3932–3937.

Chen, R., Rubanova, Y., Bettencourt, J., Duvenaud, D., 2018. Neural ordinary differential equations. arXiv:1806.07366.

Choi, J., Bracco, A., Barkan, R., Shchepetkin, A.F., McWilliams, J.C., Molemaker, J.M., 2017. Submesoscale dynamics in the northern Gulf of Mexico. Part III: Lagrangian implications. J. Phys. Oceanogr. 47 (9), 2361–2376.

Coelho, E.F., Hogan, P., Jacobs, G., Thoppil, P., Huntley, H.S., Haus, B.K., Lipphardt, B.L., Kirwan, A.D., Ryan, E.H., Olascoaga, J., Beron-Vera, F., Poje, A.C., Griffa, A., Özgökmen, T.M., Mariano, A.J., Novelli, G., Haza, A.C., Bogucki, D., Chen, S.S., Curcic, M., Iskandarani, M., Judt, F., Laxague, N., Reniers, A.J., Valle-Levinson, A., Wei, M., 2015. Ocean current estimation using a multi-model ensemble Kalman filter during the Grand Lagrangian Deployment experiment (GLAD). Ocean Model. 87, 86–106.

Dormand, J., Prince, P., 1980. A family of embedded Runge–Kutta formulae. J. Comput. Appl. Math. 6 (1), 19–26.

Duch, W., Jankowski, N., 1999. Survey of neural transfer functions. Neural Comput. Surv. 2, 163–212.

Dueben, P.D., Bauer, P., 2018. Challenges and design choices for global weather and climate models based on machine learning. Geosci. Model Dev. 11 (10), 3999–4009.

Elman, J.L., 1990. Finding structure in time. Cogn. Sci. 14 (2), 179–211.

Elman, J.L., 1991. Distributed representations, simple recurrent networks, and grammatical structure. Mach. Learn. 7 (2), 195–225.

Elman, J.L., 1993. Learning and development in neural networks: The importance of starting small. Cognition 48 (1), 71–99.

Enriquez, C., Mariño-Tapia, I.J., Herrera-Silveira, J.A., 2010. Dispersion in the Yucatan coastal zone: Implications for red tide events. Cont. Shelf Res. 30 (2), 127–137.

Faghmous, J.H., Banerjee, A., Shekhar, S., Steinbach, M., Kumar, V., Ganguly, A.R., Samatova, N., 2014. Theory-guided data science for climate change. Computer 47 (11), 74–78.

Franz, K., Roscher, R., Milioto, A., Wenzel, S., Kusche, J., 2018. Ocean eddy identification and tracking using neural networks. Int. Geosci. Remote Sens. Symp. 6887–6890.

González, D.L., Angus, M.P., Tetteh, I.K., Bello, G.A., Padmanabhan, K., Pendse, S.V., Srinivas, S., Yu, J., Semazzi, F., Kumar, V., Samatova, N.F., 2015. On the data-driven inference of modulatory networks in climate science: An application to West African rainfall. Nonlinear Process. Geophys. 22 (1), 33–46.

Gough, M.K., Reniers, A.J.H.M., MacMahan, J.H., Howden, S.D., 2016. Resonant near-surface inertial oscillations in the northeastern Gulf of Mexico. J. Geophys. Res. Oceans 121 (4), 2163–2182.

Griffa, A., 1996. Applications of stochastic particle models to oceanographic problems. In: Stoch. Model. Phys. Oceanogr.. Birkhäuser Boston, Boston, MA, pp. 113–140.

Haza, A.C., Griffa, A., Martin, P., Molcard, A., Özgökmen, T.M., Poje, A.C., Barbanti, R., Book, J.W., Poulain, P.M., Rixen, M., Zanasca, P., 2007. Model-based directed drifter launches in the Adriatic Sea: Results from the DART experiment. Geophys. Res. Lett. 34 (10), L10605.

Haza, A.C., Özgökmen, T.M., Griffa, A., Garraffo, Z.D., Piterbarg, L., 2012. Parameterization of particle transport at submesoscales in the Gulf Stream region using Lagrangian subgridscale models. Ocean Model. 42, 31–49.

Haza, A.C., Özgökmen, T.M., Hogan, P., 2016. Impact of submesoscales on surface material distribution in a gulf of Mexico mesoscale eddy. Ocean Model. 107, 28–47.

Holland, W.R., Lin, L.B., 1975a. On the generation of mesoscale eddies and their contribution to the oceanic general circulation. I. A preliminary numerical experiment. J. Phys. Oceanogr. 5 (4), 642–657.

Holland, W.R., Lin, L.B., 1975b. On the generation of mesoscale eddies and their contribution to the oceanic general circulation. II. A parameter study. J. Phys. Oceanogr. 5 (4), 658–669.

Isaji, T., Spaulding, M.L., Allen, A.A., 2006. Stochastic particle trajectory modeling techniques for spill and search and rescue models. In: Estuar. Coast. Model.. American Society of Civil Engineers, Reston, VA, pp. 537–547.

Jordan, M.I., 1986. Serial Order: A Parallel Distributed Processing Approach. Tech. Rep. No. (8604), Univ. Calif. San Diego Inst. Cogn. Sci., pp. 1–46.

Koshel', K.V., Prants, S.V., 2006. Chaotic advection in the ocean. Phys.-Usp. 49 (11), 1151–1178.

Krasnopolsky, V.M., Chalikov, D.V., Tolman, H.L., 2002. A neural network technique to improve computational efficiency of numerical oceanic models. Ocean Model. 4 (3–4), 363–383.

Kubat, M., 1989. Floating approximation in time-varying knowledge bases. Pattern Recognit. Lett. 10 (4), 223–227.

Kubat, M., 2017. An Introduction to Machine Learning, second ed. Springer International Publishing, Cham.

Kubat, M., Holte, R.C., Matwin, S., 1998. Machine learning for the detection of oil spills in satellite radar images. Mach. Learn. 30 (2–3), 195–215.

Kuitenbrouwer, D., Reniers, A., MacMahan, J., Roth, M.K., 2018. Coastal protection by a small scale river plume against oil spills in the Northern Gulf of Mexico. Cont. Shelf Res. 163, 1–11.

Lee, S., You, D., 2017. Prediction of laminar vortex shedding over a cylinder using deep learning. arXiv:1712.07854.

Lguensat, R., Sun, M., Fablet, R., Mason, E., Tandeo, P., Chen, G., 2018. EddyNet: A deep neural network for pixel-wise classification of oceanic eddies. In: Int. Geosci. Remote Sens. Symp., Vol. 2018-July. IEEE, pp. 1764–1767.

Lumpkin, R., Özgökmen, T., Centurioni, L., 2017. Advances in the application of surface drifters. Ann. Rev. Mar. Sci. 9 (1), 59–81.

Mariano, A.J., Kourafalou, V.H., Srinivasan, A., Kang, H., Halliwell, G.R., Ryan, E.H., Roffer, M., 2011. On the modeling of the 2010 Gulf of Mexico Oil Spill. Dyn. Atmos. Ocean. 52 (1–2), 322–340.

Mariano, A.J., Ryan, E.H., Huntley, H.S., Laurindo, L., Coelho, E., Griffa, A., Özgökmen, T.M., Berta, M., Bogucki, D., Chen, S.S., Curcic, M., Drouin, K., Gough, M., Haus, B.K., Haza, A.C., Hogan, P., Iskandarani, M., Jacobs, G., Kirwan, A.D., Laxague, N., Lipphardt, B., Magaldi, M.G., Novelli, G., Reniers, A., Restrepo, J.M., Smith, C., Valle-Levinson, A., Wei, M., 2016. Statistical properties of the surface velocity field in the northern Gulf of Mexico sampled by GLAD drifters. J. Geophys. Res. Oceans 121 (7), 5193–5216.

Massoud, E.C., Huisman, J., Benincà, E., Dietze, M.C., Bouten, W., Vrugt, J.A., 2018. Probing the limits of predictability: Data assimilation of chaotic dynamics in complex food webs. Ecol. Lett. 21 (1), 93–103.

McGoogan, J., 1975. Satellite altimetry applications. IEEE Trans. Microw. Theory Tech. 23 (12), 970–978.

Mensa, J.A., Timmermans, M.L., Kozlov, I.E., Williams, W.J., Özgökmen, T.M., 2018. Surface drifter observations from the Arctic Ocean's Beaufort Sea: Evidence for submesoscale dynamics. J. Geophys. Res. Oceans 123 (4), 2635–2645.

Mohan, A.T., Gaitonde, D.V., 2018. A deep learning based approach to reduced order modeling for turbulent flow control using LSTM neural networks. arXiv: 1804.09269.

Molcard, A., Poje, A.C., Özgökmen, T.M., 2006. Directed drifter launch strategies for Lagrangian data assimilation using hyperbolic trajectories. Ocean Model. 12 (3–4), 268–289.

Moradi Kordmahalleh, M., Gorji Sefidmazgi, M., Homaifar, A., 2016. A sparse recurrent neural network for trajectory prediction of Atlantic hurricanes. In: Proc. 2016 Genet. Evol. Comput. Conf.. GECCO '16, ACM Press, New York, New York, USA, pp. 957–964.

Nielsen, M.A., 2014. Neural networks and deep learning. In: Mach. Learn.. Determination Press.

Normile, D., 2014. Lost at sea. Science 344 (6187), 963–965.

Novelli, G., Guigand, C.M., Özgökmen, T.M., 2018. Technological advances in drifters for oil transport studies. Mar. Technol. Soc. J. 52 (6), 53–61.

Olascoaga, M.J., 2010. Isolation on the West Florida shelf with implications for red tides and pollutant dispersal in the Gulf of Mexico. Nonlinear Process. Geophys. 17 (6), 685–696.

Olascoaga, M.J., Beron-Vera, F.J., Haller, G., Triñanes, J., Iskandarani, M., Coelho, E.F., Haus, B.K., Huntley, H.S., Jacobs, G., Kirwan, A.D., Lipphardt, B.L., Özgökmen, T.M., Reniers, A., Valle-Levinson, A., 2013. Drifter motion in the Gulf of Mexico constrained by altimetric Lagrangian coherent structures. Geophys. Res. Lett. 40 (23), 6171–6175.

Olascoaga, M.J., Haller, G., 2012. Forecasting sudden changes in environmental pollution patterns. Proc. Natl. Acad. Sci. 109 (13), 4738–4743.

Ouala, S., Nguyen, D., Drumetz, L., Chapron, B., Pascual, A., Collard, F., Gaultier, L., Fablet, R., 2019. Learning latent dynamics for partially-observed chaotic systems. arXiv Prepr..

Özgökmen, T., Chassignet, E., Dawson, C., Dukhovskoy, D., Jacobs, G., Ledwell, J., Garcia-Pineda, O., MadDonald, I., Morey, S., Olascoaga, M., Poje, A., Reed, M., Skancke, J., 2016. Over what area did the oil and gas spread during the 2010 Deepwater Horizon oil spill? Oceanography 29 (3), 96–107.

Özgökmen, T.M., Griffa, A., Mariano, A.J., 2000. On the predictability of Lagrangian trajectories in the ocean. J. Atmos. Ocean. Technol. 17, 366–383.

Özgökmen, T.M., Iliescu, T., Fischer, P.F., 2009. Large eddy simulation of stratified mixing in a three-dimensional lock-exchange system. Ocean Model. 26 (3–4), 134–155.

Özgökmen, T.M., Piterbarg, L.I., Mariano, A.J., Ryan, E.H., 2001. Predictability of drifter trajectories in the Tropical Pacific Ocean. J. Phys. Oceanogr. 31 (9), 2691–2720.

Özgökmen, T.M., Poje, A.C., Fischer, P.F., Childs, H., Krishnan, H., Garth, C., Haza, A.C., Ryan, E., 2012. On multi-scale dispersion under the influence of surface mixed layer instabilities and deep flows. Ocean Model. 56, 16–30.

Özgökmen, T.M., Poje, A.C., Fischer, P.F., Haza, A.C., 2011. Large eddy simulations of mixed layer instabilities and sampling strategies. Ocean Model. 39 (3–4), 311–331.

Payeur, P., Le-Huy, H., Gosselin, C., 1995. Trajectory prediction for moving objects using artificial neural networks. IEEE Trans. Ind. Electron. 42 (2), 147–158.

Peacock, T., Haller, G., 2013. Lagrangian coherent structures: The hidden skeleton of fluid flows. Phys. Today 66 (2), 41–47.

Poje, A.C., Ozgökmen, T.M., Lipphardt, B.L., Haus, B.K., Ryan, E.H., Haza, A.C., Jacobs, G.A., Reniers, A.J.H.M., Olascoaga, M.J., Novelli, G., Griffa, A., Beron-Vera, F.J., Chen, S.S., Coelho, E., Hogan, P.J., Kirwan, A.D., Huntley, H.S., Mariano, A.J., 2014. Submesoscale dispersion in the vicinity of the Deepwater Horizon spill. Proc. Natl. Acad. Sci. USA 111 (35), 12693–12698.

Prasad, T.G., Hogan, P.J., 2007. Upper-ocean response to Hurricane Ivan in a 1/25° nested Gulf of Mexico HYCOM. J. Geophys. Res. 112 (C4), C04013.

Rixen, M., Ferreira-Coelho, E., Signell, R., 2008. Surface drift prediction in the Adriatic Sea using hyper-ensemble statistics on atmospheric, ocean and wave models: Uncertainties and probability distribution areas. J. Mar. Syst. 69 (1–2), 86–98.

Roarty, H., Glenn, S., Kohut, J., Gong, D., Handel, E., Rivera, E., Garner, T., Atkinson, L., Brown, W., Jakubiak, C., Muglia, M., Haines, S., Seim, H., 2010. Operation and application of a regional high-frequency radar network in the Mid-Atlantic Bight. Mar. Technol. Soc. J. 44 (6), 133–145.

Rupolo, V., 2007. A Lagrangian-based approach for determining trajectories taxonomy and turbulence regimes. J. Phys. Oceanogr. 37 (6), 1584–1609.

Schroeder, K., Chiggiato, J., Haza, A.C., Griffa, A., Özgökmen, T.M., Zanasca, P., Molcard, A., Borghini, M., Poulain, P.M., Gerin, R., Zambianchi, E., Falco, P., Trees, C., 2012. Targeted Lagrangian sampling of submesoscale dispersion at a coastal frontal zone. Geophys. Res. Lett. 39 (11).

Shay, L.K., Lentz, S.J., Graber, H.C., Haus, B.K., 1998. Current structure variations detected by high-frequency radar and vector-measuring current meters. J. Atmos. Ocean. Technol. 15 (1), 237–256.

Smith, K.M., Hamlington, P.E., Fox-Kemper, B., 2015. Effects of submesoscale turbulence on ocean tracers. J. Geophys. Res. Oceans 121 (1), 908–933.

Solomatine, D., See, L., Abrahart, R., 2008. Data-driven modelling: Concepts, approaches and experiences. In: Pract. Hydroinformatics. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 17–30.

Stommel, H., 1948. The westward intensification of wind-driven ocean currents. Trans. Am. Geophys. Union 29 (2), 202–206.

Taylor, J.R., 2018. Accumulation and subduction of buoyant material at submesoscale fronts. J. Phys. Oceanogr. 48 (6), 1233–1241.

Tolman, H.L., Krasnopolsky, V.M., Chalikov, D.V., 2005. Neural network approximations for nonlinear interactions in wind wave spectra: Direct mapping for wind seas in deep water. Ocean Model. 8 (3), 253–278.

Toms, B.A., Kashinath, K., Prabhat, Yang, D., 2020. Testing the reliability of interpretable neural networks in geoscience using the Madden-Julian Oscillation. Geosci. Model Dev. Discussions 2020, 1–22.

Touretzky, D.S., Pomerleau, D.A., 1989. What's hidden in the hidden layers? Byte 14 (8), 227–233.

Wei, M., Jacobs, G., Rowley, C., Barron, C.N., Hogan, P., Spence, P., Smedstad, O.M., Martin, P., Muscarella, P., Coelho, E., 2016. The performance of the US Navy's RELO ensemble, NCOM, HYCOM during the period of GLAD at-sea experiment in the Gulf of Mexico. Deep-Sea Res. II 129, 374–393.

Wikner, A., Pathak, J., Hunt, B., Girvan, M., Arcomano, T., Szunyogh, I., Pomerance, A., Ott, E., 2020. Combining machine learning with knowledge-based modeling for scalable forecasting and subgrid-scale closure of large, complex, spatiotemporal systems. Chaos Interdiscip. J. Nonlinear Sci. 30 (5), 053111.

Yang, H., Liu, Z., 1997. The three-dimensional chaotic transport and the great ocean barrier. J. Phys. Oceanogr. 27 (7), 1258–1273.