A DNN Framework for Learning Lagrangian Drift With Uncertainty

Joseph R. Jenkins, Adeline Paiement, Yann Ourmières, Julien Le Sommer, Jacques Verron, Clément Ubelmann, Hervé Glotin

A DNN Framework for Learning Lagrangian Drift With Uncertainty

Joseph Jenkins[1,2,3,5*], Adeline Paiement[2], Yann Ourmières[3], Julien Le Sommer[4], Jacques Verron[1], Clément Ubelmann[5] and Hervé Glotin[2]

[1]Ocean Next.
[2]Université de Toulon, Aix Marseille Univ, CNRS, LIS, Marseille, France.
[3]Mediterranean Institute of Oceanography, Université de Toulon, Aix Marseille Univ, CNRS, IRD, MIO, Toulon, France.
[4]Univ. Grenoble Alpes, CNRS, IRD, Grenoble INP, IGE, 38000 Grenoble, France.
[5]Datlas.

*Corresponding author(s). E-mail(s): joseph.jenkins@lis-lab.fr;

Abstract

Reconstructions of Lagrangian drift, for example for objects lost at sea, are often uncertain due to unresolved physical phenomena within the data. Uncertainty is usually overcome by introducing stochasticity into the drift, but this approach requires specific assumptions for modelling uncertainty. We remove this constraint by presenting a purely data-driven framework for modelling probabilistic drift in flexible environments. We train a CNN to predict the temporal evolution of probability density maps of particle locations from $\boldsymbol{t}$ to $\boldsymbol{t+1}$ given an input velocity field. We generate groundtruth density maps on the basis of ocean circulation model simulations by simulating uncertainty in the initial position of particle trajectories. Several loss functions for regressing the predicted density maps are tested. Through evaluating our model on unseen velocities from a different year, we find its outputs to be in good agreement with numerical simulations, suggesting satisfactory generalisation to different dynamical situations.

Keywords: Uncertainty representations, Pixel-wise regression, Flow modelling, Physical science

1 Introduction

We present a data-driven framework for learning Lagrangian drift over a given timestep (e.g. one day in our experiments) in the presence of uncertainty. Uncertainty arises when dynamical systems cannot be perfectly described due to imperfect modelling of either the dynamics or the system state. Modelling capabilities are constrained by availability of compute, knowledge of the underlying physical phenomena (particularly at small scales), and sensor resolution. Meanwhile, chaotic systems result in minor state variations to have significant and unpredictable influences in the observed behaviour. Thus, modelling drift with uncertainty is critical for many applications whose dynamics are chaotic or whose inputs (e.g. velocity field) do not sufficiently resolve the necessary dynamics. This includes applications such as ocean and atmospheric dynamics [1].

Uncertainty is usually modelled through stochastic differential equations (SDEs) to account for uncaptured physical phenomena at small scales [2]. If the phenomena can be captured such that the main source of uncertainty comes from the effects of chaos, accounting for uncertainty may be simplified through deterministic sampling of particle drifts with random variations in the initial conditions e.g. [3]. However, in practice, forecasting applications are constrained by low resolution models that fail to capture the necessary physical phenomena and as such must be accounted for e.g. through SDEs.

We follow a different approach to account for uncertainty in the drift which does not rely on resolving physics equations. We use a deep neural network (DNN) for modelling the drift (Section 4.2), and we propose a probabilistic representation of particle location for representing uncertainty (Section 4.1). Through using this representation, our DNN inherently includes the concept of uncertainty in its internal modelling. While we demonstrate our approach using simulated drifts where uncertainty is produced by sampling with respect to the initial position of particles as in [3], our framework is more general and may be trained with non-simulated drifts or different ways to produce uncertainty in the future.

Our simulations are performed on realistic, high-resolution oceanic surface currents representative of real-world past ocean states in the north-west Mediterranean sea. Thus, we demonstrate our framework by learning a drift model of floating objects at sea. As our framework is completely data-driven, it supports a great deal of flexibility in what it can take in as input. Any information representative of surface currents such as velocity fields or sea surface height (SSH) measures may be used. Even if this information does not capture some of the physical phenomena, a data-driven model may be able to infer the missing phenomena provided they are captured within the training examples. We believe these to be considerable advantages compared to traditional physics equations–based modelling approaches that lack the flexibility to extract information from different physical quantities and resolutions.

Our contributions may be summarised as: 1) We propose a new approach to modelling Lagrangian drift with uncertainty based on deep learning. Our

framework is applicable to observations representative of any source (simulated or not) or measure of uncertainty. 2) This approach is supported by a new statistical representation of particle location for modelling uncertain drift. To the best of our knowledge, no previous framework used a probabilistic location in the modelling of Lagrangian drift. 3) We demonstrate our method on a new dataset of simulated drifts of floating objects at sea characterised by uncertainty in the objects' initial positions, which will be released together with this paper.

The rest of this article is organised as follows. Section 2 reviews previous works on uncertain drift and trajectory modelling. Section 3 introduces our dataset and Section 4 presents our method. Experimental results are discussed in Section 5. Section 6 concludes the paper.

2 Previous works

Uncertainty in Lagrangian drift is usually modelled using stochastic trajectories through SDEs. Stochasticity may be used to parameterise unresolved physics at subgrid scales, either by formulating the SDE as a Fokker-Planck equation [4] or by fitting an SDE to simulated stochastic trajectories [5]. For the application of sea surface currents, examples of unresolved physics are the motions of eddies, waves, or small-scale turbulence. For a review on how to use SDEs to account for oceanic phenomena, see [2]. Stochastic trajectories may be simulated by the means of randomly varying a particle's displacement, velocity, or its acceleration. Contrary to a pure data-driven approach, SDEs may not be able to describe arbitrary sources of uncertainty. Furthermore, they are limited in their reliance on specific physical quantities such as velocity information.

Previous works utilising machine learning to predict Lagrangian drift aim to model drift deterministically rather than probabilistically. [6, 7] train their prediction models on individual instances of artificial simulated flows, and as such they do not consider generalisation to different flows (e.g. real spatio-temporal conditions). Instead of using simulations, [8, 9] learn from past observations of drifters at sea. [8] used a neural network to predict drifter displacement from wind and flow velocity, while [9] solve physics equations with a neural network implementing an additional term to learn the unknown physical phenomena. In doing so, [9] demonstrated an ability to model drift behaviour that was not described by a baseline physics model. However, such modelling capabilities diminished as the spatio-temporal conditions deviated from the training set, indicating a lack of generalisation to different flows. Due to the limited availability of past observations combined with the passive nature of their drift in real-world environments, the resulting coverage of conditions is typically insufficient to meaningfully represent the true distribution. [10] considers the case of learning to model drift of a scalar field as opposed to individual particles. However, their methodology is tightly integrated with the

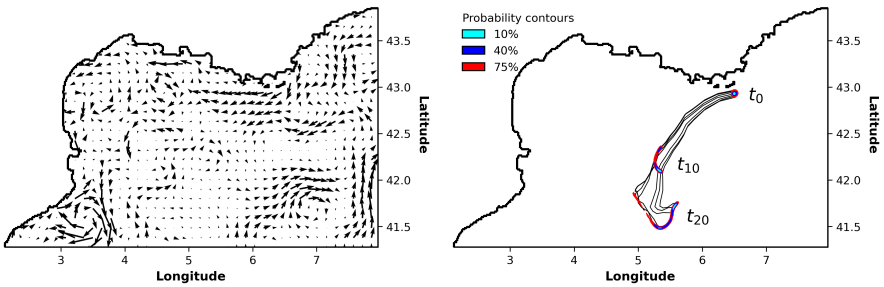

Fig. 1 Overview of data. Left) Example velocity field (downgraded resolution for visualisation). Right) Overlay of 3 probability density snapshots (shown as filled probability contours). Example trajectories used to estimate the probability density snapshots are shown in black.

equation for advecting concentration fields, which is fundamentally different to advecting particles with uncertainty.

3 Data

We introduce a new dataset comprised of 1) velocity fields of sea surface currents (Figure 1 left and Section 3.1) and 2) 2D probabilistic snapshots of drifting particle positions (Figure 1 right and Section 3.2). Our snapshots are produced from Lagrangian drift simulations on our velocity fields, which in turn are produced from realistic high resolution numerical models validated by real observations.

In this study, we model drift over a one-day timestep, so we prepare our velocity fields and snapshots to have a timestep of one day. This timestep is motivated by the spatial resolution of our velocity fields and the dynamics of the region considered in our study. We observe a notable but small enough amount of drift between timesteps, which is important for providing a meaningful learning criteria.

3.1 Velocity fields of surface currents

We use surface currents from the GLAZUR64 [11] ocean general circulation model (OGCM) which is based on the NEMO [12] OGCM. The model has been validated with real observational data (current meters and sea surface temperature/height) [11, 13] in order to provide high-resolution realistic snapshots of past ocean states within the north-west Mediterranean sea (lon 2–8° E, lat 41.3–43.9° N). In this study, we consider the surface to be two-dimensional by utilising the uppermost layer only. In the ocean modelling community, it is standard to approximate the drift of floating objects by ignoring depth information [14]. The two-dimensional resolution is $\frac{1}{64}$° which equates to each grid

cell (pixel)[1] being representative of ∼1.3×1.3 km. GLAZUR64 produces an output every minute, so we average its outputs over a one-day period in order to fulfil our one-day scenario.

The surface currents are composed of two velocity components each represented by a matrix over the spatial domain: U (zonal component of the flow) and V (meridional component). NEMO uses a staggered grid to represent velocity components such that U and V are offset by half a grid cell to the right and down, respectively. In preparation for Section 4.2 where we provide the velocity components as input into a CNN, we align the components to the pixel centres using linear interpolation. We also replace the NaN values from land pixels to 0 which provides a natural interpretation of zero flow to the CNN. An example of the final velocity field is illustrated in Figure 1 left.

In Section 3.2, we simulate particle trajectories using velocity fields for the years 2018 and 2016. We sample from complete years in order to representatively capture seasonal variances in the currents. In Section 5, we use the data from 2018 for both training and testing. As a result, velocity fields from the same season, or even the same day, may be used for both training and testing[2]. Data from 2016 is reserved exclusively for testing in order to provide a more thorough evaluation of our model's generalisation to unseen velocities and drifts. The dynamics of sea surface currents are not correlated across years, so we expect the evaluation for 2016 to be representative of the performance for other years.

3.2 Particle trajectories

Using the daily velocity fields of surface currents from GLAZUR64 (Section 3.1), we simulate 30-day trajectories of floating particles and divide them up into daily snapshots. We work with *probabilistic trajectories*, meaning that for each snapshot, the particle's position is not represented by a deterministic 2D point, but rather by a 2D probability distribution (see Figure 1 right). We define the representation for this distribution in Section 4.1.

Despite the methodological premise of our work being to predict Lagrangian drift across a single timestep, we generate long trajectories, from which we extract snapshots, with the purpose of introducing variance into the level of uncertainty of the snapshots. As a particle traverses over time, the uncertainty in its position will grow as the potential for it to take different paths increases (see Figure 1 right).

3.2.1 Particle advection

We use the OceanParcels [15] library to advect massless, floating particles on our velocity fields using a 4th order Runge-Kutta integration scheme, where

[1] Although NEMO uses curvilinear coordinates, the limited region that we consider makes it reasonable to neglect projection errors and to associate each grid cell to a Cartesian pixel.

[2] However, different drifter trajectories (thus different input locations) are considered to avoid the DNN relying on memory for drift prediction (see Section 3.2).

we update the state of the particles every six hours. The positioning of particles is continuous, so OceanParcels performs space-time interpolation of the discretised velocity fields.

We advect particles for up to 30 days and save their positions daily. Particles may not always complete a full 30-day trajectory due to interactions at the boundaries. There are two types of boundaries: the ocean-land boundary and the open boundary (see Figure 1). The open boundary is named as such due to being caused by the cutoff of our data coverage such that the neighbouring ocean values are unknown. We define two conditions for the premature termination of particles: 1) an advection step has caused a particle to escape the ocean-land or open boundary, or 2) a particle has made contact with an ocean cell (pixel) at the open boundary. The second condition exists to prevent particles from getting stuck and accumulating at the open boundary.

3.2.2 Probabilistic trajectories

As discussed in Section 2, introducing random behaviour into the modelling of Lagrangian drift serves the purpose of accounting for uncertainty within the data or drift process. Thus, we generate trajectories whose drifts are probabilistically defined by some chosen source of randomness. Our probabilistic trajectories represent a particle's position as a probability distribution rather than a single point. We approximate this distribution by advecting N_P particles, where we empirically decide $N_P = 10,000$ as being sufficient for approximating the distribution. To demonstrate our framework, we randomly perturb the initial position of each particle within a 5 km radius[3]. This choice of randomness is motivated by being simplistic and efficient, as it allows for the advection process to remain completely deterministic. In practice, our methodology could be applied to any desired source of randomness such as perturbations in a drifting particle's velocity or position.

3.2.3 Snapshots of probabilistic trajectories

In preparation for training our CNN to predict Lagrangian drift over a given timestep (one day in our experiments), we divide the probabilistic trajectories into snapshots. Each snapshot represents a probability distribution of a particle's position in space. As we approximate this distribution by advecting N_P particles, our snapshots are initially represented as a group of particles, before being converted into probability density maps in Section 4.1. As mentioned previously, particle advection may be prevented at the boundaries, thus the sum of a snapshot's distribution may be less than one due to the number of particles in a snapshot being less than N_P.

[3]If this results in particles to lie outside of the ocean's domain then we discard them, and hence the actual number of particles may be less than N_P.

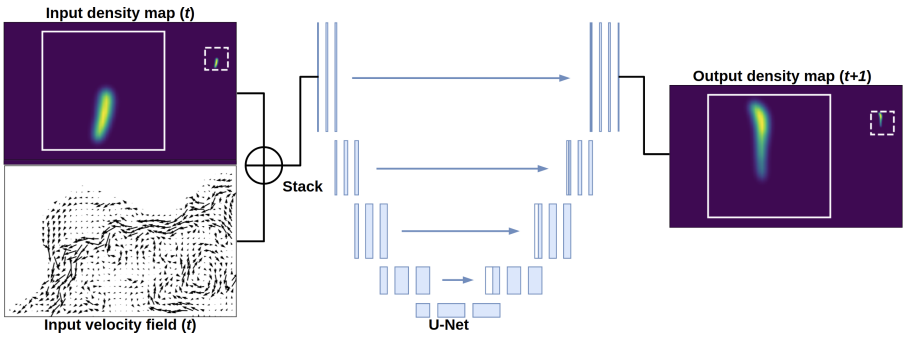

Fig. 2 Overview of methodology. The input density map $D_t^{M\times N}$ is stacked with the corresponding velocity field $U_t^{2\times M\times N}$ to give the input matrix $X_t^{3\times M\times N}$, which is fed into a U-Net architecture to output the successive density map snapshot $D_{t+1}^{M\times N}$. We show zoomed overlays (solid white square) of the foreground region of the density map snapshots (dashed white square). Note that the coastline is not visible in the density maps due to setting the land values to have probability density values of 0.

3.2.4 Deployment of probabilistic trajectories

For the year 2018, we deploy N_T 30-day probabilistic trajectories whose initial positions are randomly sampled over the ocean's spatio-temporal domain. We choose $N_T = 10,000$ to encourage a wide spatio-temporal coverage of our simulated trajectories. To ensure a full 30-day trajectory can be completed, we set the last sampling date to December 1st. To evaluate our model on a different year, we deploy $0.15N_T$ trajectories for the year 2016 with the same sampling characteristics as the year 2018.

4 Methodology

We train a CNN (Section 4.2) to predict the one-day evolution of particle drift with uncertainty. It takes as input a velocity field (Section 3.1) and a probability density map (Section 4.1) of particle location. An overview of this process is shown in Figure 2. We train the CNN by regressing the pixel-wise probability density values between the groundtruth and predicted density maps at $t+1$ (one day). We propose and compare three different loss functions in Sections 4.2 and 5.1.

4.1 Probability density maps for uncertainty representation

In Section 3.2, we approximate the distribution of uncertain particle drift by advecting a large number of particles. Here, we choose to represent the 2D probability distribution of particle location in the form of probability density maps. This choice is motivated by the ability to homogenise the representations between probability distribution and velocity field, which are both given as input into a CNN in Section 4.2.

To create our density maps, we produce a first approximation of the probability distribution by computing a 2D histogram of the locations of the snapshot's N_p particles (see Section 3.2) with respect to the grid of our velocity fields. To account for only having a finite number of particles, we produce a more plausible approximation of the distribution by applying a Gaussian filter ($\sigma = 1$) to the histogram in order to smooth out its unnaturally sharp spatial gradients. We note that the sum of our density maps may not always sum to one due to the possibility of particles escaping the region, as explained in Section 3.2.1.

4.2 Neural network

4.2.1 Architecture

Our CNN takes as input a 3-channel matrix consisting of a 2-channel (U and V) velocity field (Section 3.1) stacked with a 1-channel probability density map (Section 4.1) of particle location. It regresses the probability density map representation defined in Section 4.1. To demonstrate our methodology, we use the popular U-Net [16] architecture. U-Net's encoder-decoder architecture is well suited for pixel-wise regression due to its deconvolutional layers which map low-dimensional feature encodings back into the original image space. Skip connections are also used to preserve higher-dimensional representations of spatial information at different scales. In practice, other architectures, e.g. designed for segmentation, can be applied to our pixel-wise regression setup e.g. [17] and [18].

4.2.2 Loss function

We consider three loss functions to evaluate the quality of the model's predicted density map $\hat{D}$ with respect to the reference density map D^{t+1}. $L_{position}$ minimises the MSE between the predicted and groundtruth probability density maps of position (Eq. (1)). L_{drift} evaluates MSE on the drift between t and $t+1$ (Eq. (2)). $L_{threshold}$ is similar to $L_{position}$ but it evaluates MSE on the foreground pixels (i.e. non-zero) of $\hat{D}$ and D^{t+1} (Eq. (3)). As we know that particles can only exist within the set of ocean pixels $\mathcal{O}$, we ignore any land pixels in the output, including when computing the loss.

$$L_{position} = \frac{1}{|\mathcal{O}|} \sum_{x \in \mathcal{O}} (D_x^{t+1} - \hat{D}_x)^2 \tag{1}$$

$$L_{drift} = \frac{1}{|\mathcal{O}|} \sum_{x \in \mathcal{O}} (R_x - \hat{R}_x)^2 \tag{2}$$

$$L_{threshold} = \frac{1}{|\mathcal{O}_\tau|} \sum_{x \in \mathcal{O}_\tau} (D_x^{t+1} - \hat{D}_x)^2 \tag{3}$$

where $R = D^{t+1} - D^t$, $\hat{R} = \hat{D} - D^t$, and $\mathcal{O}_\tau = \{x \in \mathcal{O} \mid (D_x^{t+1} > 0) \vee (\hat{D}_x > 0)\}$.

As we aim to learn the change incurred to a density map by a drift, the drift loss L_{drift} provides a learning criteria that matches these aims directly. This helps to normalise the learning criteria across varying levels of density gradients, as $L_{position}$ may encourage the model to learn the identity function in response to small gradients. During inference, we add the prediction of the drift map $\hat{R}$ to the input D^t to recover the density map $\hat{D}$.

The sparse nature of our density maps motivates the use of the thresholded loss $L_{threshold}$ for focusing the learning on the scarce foreground information as opposed to the abundant background. By using a hard threshold of 0 to identify the foreground, the model is encouraged to implicitly discriminate between background and foreground by predicting negative values for the background. During inference, we clip negative values to zero[4]. Thanks to the smoothing operation performed on our density maps (see Section 4.1), there is a smooth transition from foreground to background. Therefore, thresholding does not introduce any strong gradients in the density maps.

5 Experiments

For each experiment, we train four models with different seeds (0–3) and present results as 'mean (std)'. We use the Adam [19] optimiser with betas (0.9, 0.999), no weight decay, and a learning rate of 0.0001. We decay the learning rate by a factor of 10 when the validation loss has not improved for 3 epochs. We stop training prior to the second learning rate decay. We use a batch size of 16. Prior to extracting snapshots from our probabilistic trajectories, we randomly split the trajectories into training, validation, and test sets with a ratio of 70/15/15.

5.1 Comparison of loss functions

Results for the three loss functions described in Section 4 are given in Table 1. When comparing L_{drift} with $L_{position}$, we see little difference in the mean performance, although a ∼14% decrease of the mean convergence time is observed for L_{drift}. This quicker training is in line with findings of previous works (e.g. [20]) that learning residuals improves convergence. $L_{threshold}$ performs worse and stops training much earlier, indicating that the loss function has trouble converging. In future work, this may be addressed by adjusting the learning scheme. Due to its good performance and faster convergence, we would recommend L_{drift} as the loss of choice in future works. For simplicity, and since $L_{position}$ obtains similarly good results, we use it for the next experiments in this paper.

[4] As negative probability density values are not tangible, we remove them during inference for all models.

	Identity	$L_{position}$	L_{drift}	$L_{threshold}$
2018	$5{\cdot}10^{-3}$	2.76 (1.30)	2.52 (1.17)	5.57 (1.06)
2016	$5{\cdot}10^{-3}$	4.38 (0.16)	4.47 (0.13)	5.67 (1.10)
# of epochs	–	43.3 (20.1)	37.3 (13.3)	11.3 (2.38)

Table 1 MSE evaluation of loss functions (at the magnitude 10^{-8}), and number of training epochs, as 'mean (std)' over 4 runs. For reference, we show MSE of D^t and D^{t+1} (Identity).

5.2 Generalisation to different dynamics

We evaluate the ability of our models (trained on 2018 data) to generalise to different dynamical situations by testing on completely unseen data from a different year (2016). Because we use a limited number of velocity fields (365) to generate a large number of snapshots (∼300k), the input velocity fields from 2018 are the same across training, validation, and test datasets. Therefore, the results presented for 2016 are a better indicator of our model's performance as there is no bias of using the same velocity fields between training and testing sets. Surface currents do not exhibit annual trends, so we expect performances on 2016 to be representative of performances on other unseen years.

When comparing the results between different years, we observe the performance of $L_{threshold}$ to remain consistently lower than the others. $L_{position}$ and L_{drift} perform worse on average for the 2016 dataset albeit with a much tighter spread. However, in Table 1 (Identity), we verify that their predicted drifts still have a higher added value than merely approximating $\hat{D}$ as D^t. In addition, upon inspection (see Figure 3), these poorer results for 2016 come in part from a larger spread of the probability distributions, indicative of a higher uncertainty. However, the predicted drifts tend to go in the correct direction, and are visually close to groundtruth ones (even if there are more errors than for 2018). This opens promising perspectives for generalisation to different locations and times.

These results suggest that the velocity fields are well accounted for in the determination of the drift. This is further verified by a small experiment in Figure 4 where the velocity field of a random training sample of 2018 was inverted. The drift coherently happened towards the upper left direction, instead of the original down right. Thus, the input velocity field plays an essential role in the prediction of the drift. Nevertheless, the higher uncertainty for testing year 2016 indicates that an internal model of drift for the space-time location may also play a (conflicting) role in the results. Future work will need to address the balance between these two influences, in order to improve the generalisation to other dynamical situations.

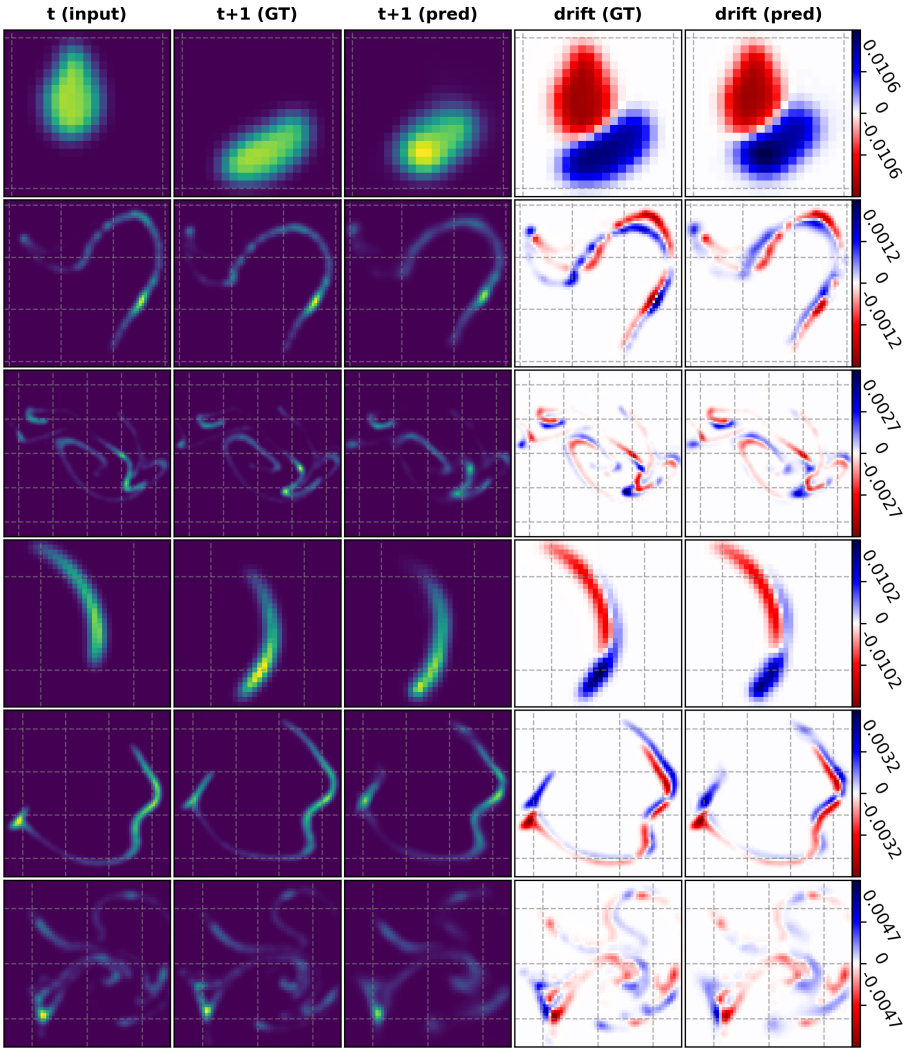

Fig. 3 Sample results for $L_{position}$ over different levels of uncertainty and different years. Left to right: D^t, D^{t+1}, $\hat{D}$, groundtruth drift R, and predicted drift $\hat{R}$. Top 3 rows: samples from 2016, bottom 3 rows: samples from 2018. Plots are zoomed on foreground regions (one grid line equals 20 pixels). Drift is from red to blue.

5.3 Application to different representations of surface currents

To demonstrate our framework's ability to adapt to different representations of surface currents, we present results for training with velocity fields of downgraded resolution (Gaussian filter of $\sigma = 1$) and SSH maps (obtained from GLAZUR64 alongside our velocity fields) using $L_{position}$. Velocity fields used for forecasting applications at sea have a much lower resolution relative to

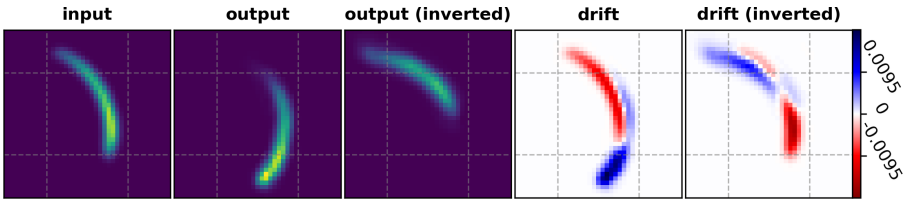

Fig. 4 Result of inverting the velocity field. Left to right: D^t, $\hat{D}$ with original and inverted velocities, predicted drift $\hat{R}$ with original and inverted velocities. $L_{position}$ is used. Note that the prediction reflects the velocity field's change in direction.

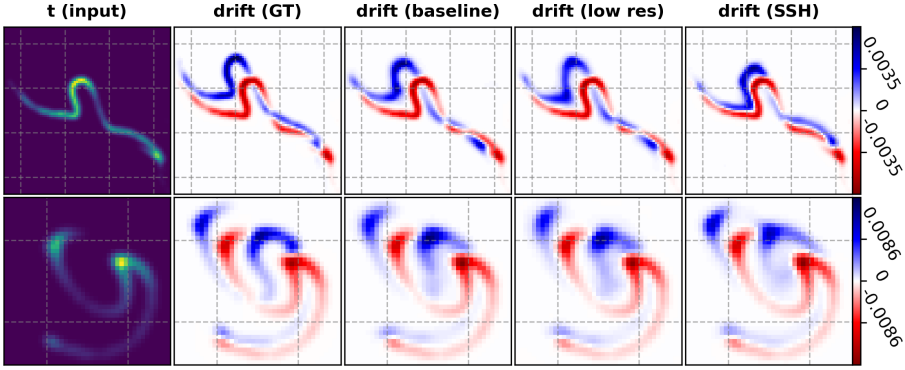

Fig. 5 Sample results for $L_{position}$ trained on different representations of surface currents. Left to right: D^t, R, $\hat{R}$ (original velocity fields), $\hat{R}$ (low res. velocity fields), and $\hat{R}$ (SSH). Top row: random sample from 2016, bottom row: random sample from 2018.

	Low-res velocity	SSH
2018	3.45 (1.09)	2.69 (0.79)
2016	4.33 (0.16)	7.27 (0.37)
# of epochs	34.5 (17.2)	46.8 (8.53)

Table 2 MSE evaluation (at the magnitude 10^{-8}) of different representations of surface currents using $L_{position}$, and number of training epochs, as 'mean (std)' over 4 runs.

research models such as the one used in this study (see Section 3.1). Meanwhile, SSH is representative of sea surface currents at large scales but does not capture small scale phenomena. However, its global availability from satellite observations makes it a very useful source of information in ocean modelling.

As illustrated in Figure 5, we observe little difference between using our high resolution velocity field, and the other two representations of surface currents. In Table 2, when comparing with Table 1, we observe that the MSE error increases only marginally for 2018 when using either low resolution representations. This indicates that the CNN is able to infer the missing physical phenomena that are not present in the input data, possibly drawing from its internal model of drift that we discussed at the end of the previous section.

For 2016, the low resolution velocity fields also obtain consistent results, while the CNN seems to struggle more with SSH, with MSE error rising from 4.38(0.16) to 7.27(0.37). This may be interpreted as the inference of missing phenomena from SSH being more dependent on the given location and time. Therefore, although the results still seem visually plausible, future users may need to keep in mind that generalisation to new dynamical situations are harder for SSH.

6 Conclusion

We addressed the problem of modelling Lagrangian drift under the influence of uncertainty by leveraging the flexibility of CNNs. We demonstrated our framework by considering an application of sea surface currents in which we modelled the uncertain drift of floating objects. We generated training data by simulating a large number of trajectories to approximate the temporal evolution of a drifting particle's probability distribution in 2D space. Our simulations were performed on velocity field representations of surface currents produced by a realistic high resolution ocean model. We found that drifts predicted by our trained model were in good agreement with the simulations, and we found our model to generalise reasonably well to different dynamical situations of a different year. We also evaluated our framework's ability to compensate for missing physical phenomena in the input data by testing on surface currents of a low resolution velocity field and SSH maps. On this harder scenario, the CNN was able to infer some of the missing information from its internal modelling of drift, and even to apply this model to different dynamical situations with reasonable success. In future works, our framework may be extended to model full drift trajectories rather than single timesteps.

Author Contributions. Conceptualization: Y.O., A.P., J.J., J.S. and H.G.; Methodology: J.J., A.P., Y.O. and J.S.; Data curation: J.J. and Y.O.; Investigation: J.J.; Software & validation: J.J.; Visualization: J.J. and A.P.; Writing - original draft: J.J. and A.P.; Writing - review & editing: A.P., Y.O. and J.S.; Supervision: A.P., Y.O. and H.G.; Project administration: H.G. and J.V.; Funding acquisition: H.G., J.V., C.U and Y.O.

Funding. This work has been supported by Ocean Next, Datlas, and ANRT by means of a PhD CIFRE grant attributed to Joseph Jenkins. It has been co-granted by the FEDER MARITTIMO GIAS (Geolocalisation by AI for maritime Security). Funding was received from Chaire Intelligence Artificielle ADSIL ANR-20-CHIA-0014 and ANR-18-CE40-0014 SMILES. The NEMO calculations were performed using GENCI-IDRIS resources, grant A0110101707.

Availability of data and materials. Data will be made public on publication.

Code availability. Code will be made public on publication.

Declarations

Competing interests. The authors declare no competing interests.

References

[1] Gill, A.E.: Atmosphere-ocean dynamics. Academic Press **30** (1982)

[2] Van Sebille, E., Griffies, S.M., Abernathey, R., Adams, T.P., Berloff, P., Biastoch, A., Blanke, B., Chassignet, E.P., Cheng, Y., Cotter, C.J., *et al.*: Lagrangian ocean analysis: Fundamentals and practices. Ocean Modelling **121**, 49–75 (2018)

[3] Koszalka, I.M., Haine, T.W.N., Magaldi, M.G.: Fates and travel times of Denmark Strait overflow water in the Irminger Basin. Journal of Physical Oceanography **43**(12), 2611–2628 (2013)

[4] Visser, A.W.: Lagrangian modelling of plankton motion: From deceptively simple random walks to fokker–planck and back again. Journal of Marine Systems **70**(3-4), 287–299 (2008)

[5] LaCasce, J.: Statistics from lagrangian observations. Progress in Oceanography **77**(1), 1–29 (2008)

[6] Han, M., Sane, S., Johnson, C.R.: Exploratory Lagrangian-based particle tracing using deep learning. arXiv preprint arXiv:2110.08338 (2021)

[7] Grossi, M.D., Kubat, M., Özgökmen, T.M.: Predicting particle trajectories in oceanic flows using artificial neural networks. Ocean Modelling **156**, 101707 (2020)

[8] Nam, Y.-W., Cho, H.-Y., Kim, D.-Y., Moon, S.-H., Kim, Y.-H.: An improvement on estimated drifter tracking through machine learning and evolutionary search. Applied Sciences **10**(22), 8123 (2020)

[9] Aksamit, N.O., Sapsis, T., Haller, G.: Machine-learning mesoscale and submesoscale surface dynamics from lagrangian ocean drifter trajectories. Journal of Physical Oceanography **50**(5), 1179–1196 (2020)

[10] Zhuang, J., Kochkov, D., Bar-Sinai, Y., Brenner, M.P., Hoyer, S.: Learned discretizations for passive scalar advection in a two-dimensional turbulent flow. Physical Review Fluids **6**(6), 064605 (2021)

[11] Ourmières, Y., Zakardjian, B., Béranger, K., Langlais, C.: Assessment of a NEMO-based downscaling experiment for the North-Western Mediterranean region: Impacts on the Northern Current and comparison with ADCP data and altimetry products. Ocean Modelling **39**(3-4), 386–404

(2011)

[12] Gurvan, M., Bourdallé-Badie, R., Chanut, J., Clementi, E., Coward, A., Ethé, C., Iovino, D., Lea, D., Lévy, C., Lovato, T., Martin, N., Masson, S., Mocavero, S., et al.: NEMO Ocean Engine. (2017). https://doi.org/10.5281/zenodo.1464816

[13] Guihou, K., Marmain, J., Ourmieres, Y., Molcard, A., Zakardjian, B., Forget, P.: A case study of the mesoscale dynamics in the North-Western Mediterranean Sea: a combined data–model approach. Ocean Dynamics **63**(7), 793–808 (2013)

[14] Mansui, J., Molcard, A., Ourmières, Y.: Modelling the transport and accumulation of floating marine debris in the Mediterranean basin. Marine pollution bulletin **91**(1), 249–257 (2015)

[15] Delandmeter, P., Sebille, E.v.: The Parcels v2.0 Lagrangian framework: new field interpolation schemes. Geoscientific Model Development **12**(8), 3571–3584 (2019)

[16] Ronneberger, O., Fischer, P., Brox, T.: U-Net: Convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention, pp. 234–241 (2015)

[17] Chen, L.-C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587 (2017)

[18] He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: International Conference on Computer Vision, pp. 2961–2969 (2017)

[19] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

[20] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)