



HAL
open science

A DNN Framework for Learning Lagrangian Drift With Uncertainty

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

► **To cite this version:**

Joseph Jenkins, Adeline Paiement, Yann Ourmières, Julien Le Sommer, Jacques Verron, et al.. A DNN Framework for Learning Lagrangian Drift With Uncertainty. 2023. hal-04106363

HAL Id: hal-04106363

<https://hal.science/hal-04106363>

Preprint submitted on 25 May 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A DNN Framework for Learning Lagrangian Drift With Uncertainty

Joseph Jenkins^{1,2,3*}, Adeline Paiement², Yann Ourmières³, Julien Le Sommer⁴, Jacques Verron⁵, Clément Ubelmann¹ and Hervé Glotin²

¹Datlas, France.

²Université de Toulon, Aix Marseille Univ, CNRS, LIS, Marseille, France.

³Mediterranean Institute of Oceanography, Université de Toulon, Aix Marseille Univ, CNRS, IRD, MIO, Toulon, France.

⁴Univ. Grenoble Alpes, CNRS, IRD, Grenoble INP, IGE, 38000 Grenoble, France.

⁵Hydro Matters, France.

*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. Using ocean circulation model simulations, we generate probabilistic trajectories of object location by simulating uncertainty in the initial object position. We train an emulator of probabilistic drift over one day given perfectly known velocities and observe good agreement with numerical simulations. Several loss functions are tested. Then, we strain our framework by training models where the input information is imperfect. On these harder scenarios, we observe reasonable predictions although the effects of data drift become noticeable when evaluating the models against unseen flow scenarios. Source code and data is available at <https://github.com/JenkinsJR/UncertainDrift>.

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 of producing uncertainty.

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 equation-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 the possibility of modelling Lagrangian drift in scenarios that cannot be solved explicitly through physics equations. (4) We show that performance can be improved by better aligning the learning criteria with the problem setting.

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

2.1 Modelling uncertainty in Lagrangian drift

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.

2.2 Machine learning for Lagrangian drift

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

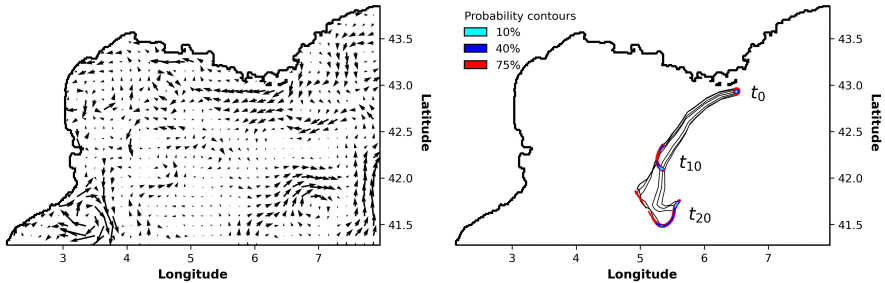


Fig. 1 Overview of data. Left) Example velocity field (downgraded resolution for visualisation). Right) Overlay of 3 probability density snapshots (shown as filled probability contours). Example trajectories used to estimate the probability density snapshots are shown in black.

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

3 Data

We generate a simulated dataset of *probabilistic trajectories* in which an object’s position is described by a 2D probability distribution as opposed to a discrete point. In practice, we achieve this by sampling many trajectories in order to estimate the statistics of the underlying uncertainty distribution. Trajectories are sampled by advecting particles on velocity fields of surface currents, which correspond to outputs from a realistic and high resolution numerical ocean model. In this study, we model drift over one day, so we discretise the velocity fields and probabilistic trajectories to one day snapshots. This timestep is motivated by the spatial resolution of the ocean model and the dynamics of the region considered in our study (north-west Mediterranean Sea).

We use modelled surface currents which represent the state of the ocean in our region of study for the years 2018 and 2016. We create one dataset of probabilistic trajectory snapshots per year, ensuring that the entire year is sampled in order to representatively capture any seasonal variances in the currents. The 2018 dataset is used for both training and evaluation¹ while 2016 is reserved exclusively for testing. This is to ensure a fair evaluation of our models’ ability to generalise to unseen flow scenarios.

¹Due to the limited number of one-day snapshots in a year, the same flow scenarios are used for both training and evaluation for the year 2018. However, trajectories are sampled across different locations (see Section 3.2) which prevents the model from relying on memory.

3.1 Velocity fields of surface currents

3.1.1 Ocean model

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 $\frac{1}{64}^\circ$ which equates to each grid cell being representative of $\sim 1.3 \times 1.3$ km.

3.1.2 Data preparation

GLAZUR64 produces an output every minute for numerical stability purposes, so we average its outputs over a one-day period in order to fulfil our one-day modelling scenario. Its velocity components (U and V) are staggered such that U and V are offset by half a grid cell down and to the right, respectively. Prior to giving these components as input to our CNN in Section 4.2, we align the components to the pixel centres using linear interpolation². We also replace the NaN values from land pixels to 0 which provides a natural interpretation of zero flow.

3.2 Probabilistic trajectories

Despite the methodological premise of our work being to model drift across a single timestep, we generate long trajectories 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 Fig. 1, right). One-day snapshots are then extracted from the trajectories to define the groundtruth drifts.

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 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 15 days and save their positions daily. Particles may not always complete a full 15-day trajectory due to interactions at the boundaries. There are two types of boundaries: the ocean-land boundary and the open boundary (see Fig. 1). The open boundary is named as such due

²Although U and V are gridded on 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.

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 Introducing uncertainty

As discussed in Section 2.1, introducing random behaviour into the modelling of Lagrangian drift serves the purpose of accounting for uncertainty within the data or drift process. Thus, we can estimate the probabilistic nature of the drift conditioned by the underlying uncertainty by sampling many (N_P) trajectories that draw from the random distribution. To demonstrate our framework, we sample trajectories whose initial positions draw from a uniform random distribution within a 5 km radius³. This choice of randomness is motivated by being simplistic and efficient, as it allows for the advection process to remain completely deterministic. For this type of uncertainty, we empirically observe $N_P=10,000$ as being sufficient for approximating the distribution. 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 Temporal snapshots of probabilistic trajectories

In preparation for training our CNN to model probabilistic drift over a given timestep (one day in our experiments), we divide the probabilistic trajectories into temporal 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 1 due to the number of particles in a snapshot being less than N_P .

3.2.4 Deployment of probabilistic trajectories

For each dataset, we deploy N_T 15-day probabilistic trajectories whose initial positions are randomly sampled over the ocean’s spatio-temporal domain. We choose $N_T=20,000$ to encourage a wide spatio-temporal coverage of our simulated trajectories. The probabilistic trajectories are split into training⁴, validation, and test sets with a ratio of 70/15/15 prior to discretising them into snapshots to ensure no cross contamination across the sets. To ensure a full 15-day trajectory can be completed, we set the last deployment date to December 16th.

³If 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 .

⁴The training split for the 2016 dataset is not used in this study.

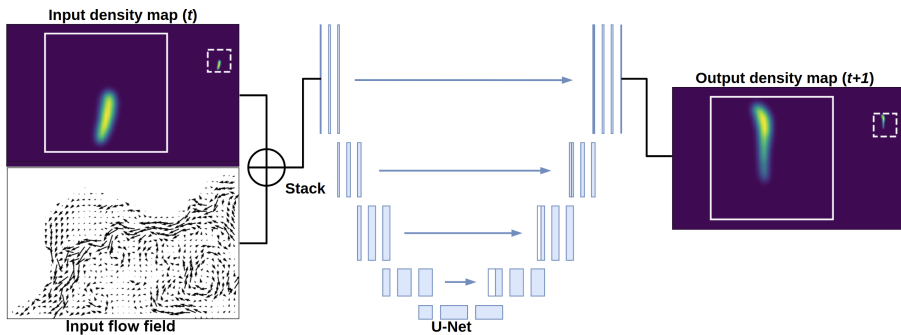


Fig. 2 Overview of methodology. The input density map at time t is stacked with the input flow field, which is fed into a U-Net architecture to output the successive density map snapshot at $t+1$. We show zoomed overlays (solid white square) of the foreground region of the density map snapshots (dashed 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.

4 Methodology

We train a U-Net based CNN to model the probabilistic one-day drift of an object’s location. The CNN takes as input a probability density map of object location as well as a representation of the underlying flow field, and outputs the density map of the following day. Our baseline model inputs the true flow field, i.e. the two consecutive velocity fields (t & $t+1$) used to simulate the drift in Section 3. Thus, our baseline model is an emulator of the simulation environment. Experiments that train models using incomplete flow fields (e.g. missing spatial or temporal information) are given in Section 5.2. All models are trained using a pixel-wise regression-based learning criteria, with various criteria considered in Section 4.2.2.

4.1 Probability density maps for uncertainty representation

In Section 3.2, we extracted temporal snapshots from probabilistic trajectories whose distributions were estimated by sampling a large number of particles. Here, we process these snapshots of particles into probability density maps by computing a 2D histogram on their locations with respect to the flow field’s grid. This allows us to homogenise the representations between the particle distribution and flow field, which are both given as input into the CNN by stacking them together (see Fig. 2). To compensate for only having a finite number of particles, we apply 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

While in practice we could use any architecture designed for the context of pixel-wise regression, such as those used for segmentation [16] and [17], we demonstrate our methodology using the popular U-Net architecture [18]. Fig. 2 shows an outline of the symmetrical encoder-decoder design of U-Net. Each block of the encoder halves its spatial dimensionality and doubles the number of feature channels, while the decoder does the opposite (using bilinear upsampling) in order to gradually recover the output in image space from low-dimensional features. Skip connections are used in the form of concatenating feature maps in order to preserve information at different spatial scales.

We use 64 convolutional channels for the first encoder block, resulting in 1024 feature maps for the final (fifth) encoder block. Unlike the original design, we normalise convolutional outputs using batch normalisation (BN) [19]. Each block of the network therefore consists of two consecutive layers of 3×3 convolution \rightarrow BN \rightarrow ReLU. Another difference is that we employ residual connections [20] between each block and use the uniform scaling rule as in [21] in order to stabilise the gradients. The final layer maps the culminating 64 feature maps into a single channel output using a 1×1 convolution. The bias of this convolution loosely implies the value of the background so we do not include it (i.e. it is frozen at zero).

4.2.2 Learning criteria

To train CNNs for pixel-wise regression tasks, the most common loss functions used are mean absolute (\mathcal{L}_1) and mean squared (\mathcal{L}_2) [22]. We try both loss functions in Section 5.1 to see which is better for our learning task. As we know that probability densities can only exist within the set of ocean pixels \mathcal{O} , we ignore any land pixels during the computation of the loss

$$\frac{1}{|\mathcal{O}|} \sum_{x \in \mathcal{O}} \mathcal{L}(D_x^{t+1}, \hat{D}_x). \quad (1)$$

As the advection equation is a partial differential equation (PDE), our learning task can also be seen through the lens of learning to solve a PDE. Given the residual nature of the solutions, reframing the learning task to explicitly learn the residual has been shown to greatly improve the convergence of optimisation [20]. In section 5.1, we experiment with learning two types of residual representations. The first uses a long additive skip connection between the input density map D^t and the output of the final 1×1 convolution layer. The loss function remains the same as Eq. 1. The second modifies the loss function to predict the residual map R , which is the difference between the target density map D^{t+1} and the input map D^t

$$\mathcal{L}^R = \mathcal{L}(R, \hat{R}), \quad (2)$$

	Zeros MSE (1e-9)	Identity MSE (1e-9)	Identity IOU ₅₀ (%)	50% contour size (px)	Distribution sum (1e-2)	Displaced density (1e-2)
2018	99.15 _{71.34}	136.52 _{115.62}	12.13 _{15.05}	63.03 _{62.44}	92.16 _{21.57}	64.85 _{23.74}
2016	100.44 _{73.26}	132.27 _{114.96}	13.36 _{15.57}	60.01 _{60.27}	90.88 _{23.36}	62.58 _{24.04}

Table 1 Groundtruth density map statistics over the 2018 and 2016 validation sets presented as mean_{std}. ‘Zeros’ assumes the prediction is all zeros and ‘Identity’ assumes the prediction is the identity function.

$$R = D^{t+1} - D^t. \quad (3)$$

4.2.3 Optimisation

We use the AdamW [23] optimiser with betas (0.9, 0.999), a weight decay of 1, and an initial learning rate of 1e-4. We decay the learning rate using a cosine annealing schedule [24] and use the linear warmup strategy as in [25], which we apply to both the learning rate and weight decay. We train our models using mixed precision [26] and a batch size of 24.

5 Experiments

For all experiments, we train three models with different seeds (0–2) and present the mean and standard deviation of the following metrics:

- **\mathcal{L}_2 Mean squared error (MSE)** – a measure related to the learning criteria which evaluates errors at the local pixel level.
- **IOU_x** – A geometrical measure which evaluates the intersection over union between the groundtruth and predicted $x\%$ probability contours. A probability contour represents the smallest distribution with a cumulative probability of $x\%$ (as in Fig. 1). We define two special cases if the total cumulative probability of the distribution is less than $x\%$. For the groundtruth, the IOU measure is undefined and not included in the statistics. For the prediction, the partial contour encompassing its limited distribution is used.
- **Mass error** – A global physics-based measure which evaluates the difference between the expected and predicted total displacement of probability density mass, defined as

$$\left| \sum_{x \in \mathcal{O}} R_x - \sum_{x \in \mathcal{O}} \hat{R}_x \right|. \quad (4)$$

5.1 Learning criteria

5.1.1 Loss function

The results in Table 2 show that \mathcal{L}_1 loss consistently performs significantly better than \mathcal{L}_2 across all metrics. We observe that \mathcal{L}_2 struggles to model lower yet important density values, indicating that the value-based exponential term

	\mathcal{L}_2 (1e-9)		IOU ₅₀ (%)		Mass error (1e-2)	
	2018	2016	2018	2016	2018	2016
\mathcal{L}_1	2.49 _{0.08}	2.52 _{0.06}	86.68 _{0.22}	86.75 _{0.17}	1.63 _{0.05}	1.59 _{0.03}
$+D_t$	2.47 _{0.07}	2.51 _{0.08}	86.78 _{0.15}	86.85 _{0.16}	1.47 _{0.03}	1.44 _{0.05}
\mathcal{L}_1^R	2.49 _{0.01}	2.54 _{0.04}	86.73 _{0.06}	86.75 _{0.03}	1.46 _{0.00}	1.44 _{0.03}
$+D_t$	2.63 _{0.05}	2.69 _{0.06}	86.43 _{0.14}	86.43 _{0.16}	1.44 _{0.01}	1.48 _{0.04}
\mathcal{L}_2	6.67 _{0.13}	6.49 _{0.11}	77.67 _{0.26}	78.24 _{0.25}	3.19 _{0.06}	3.08 _{0.04}
$+D_t$	6.89 _{0.05}	6.69 _{0.05}	77.59 _{0.09}	78.14 _{0.10}	2.89 _{0.05}	2.85 _{0.06}
\mathcal{L}_2^R	6.91 _{0.05}	6.70 _{0.06}	77.56 _{0.10}	78.13 _{0.12}	2.94 _{0.03}	2.89 _{0.03}
$+D_t$	7.37 _{0.33}	7.14 _{0.32}	76.74 _{0.51}	77.33 _{0.50}	3.08 _{0.08}	3.04 _{0.10}

Table 2 Comparison of different learning criteria evaluated on the validation sets. \mathcal{L}^R indicates regressing the residual map. $+D_t$ indicates adding the input density map to the learned output. All models are trained for 12 epochs.

is an unsuitable weighting of importance. This is coherent with the fundamental characteristic of the dataset we generate in Section 3.2 in which the level of uncertainty spreads over time. This results in the average density value to decrease as the area of importance grows larger, thereby causing \mathcal{L}_2 to favour earlier snapshots whose density values are more concentrated. The lower IOU of the 50% probability contour demonstrates that this effect is not limited to the tail end of the distribution and indeed extends to critical intervals.

We also observe that unlike \mathcal{L}_1 , \mathcal{L}_2 fails to model the background values as being exactly zero. This has the possibility of biasing the perceived error in some metrics due to the abundance of small values. Mass error, defined in Eq. 4, considers the sum over all ocean pixels and thus the totality of the background may be non-negligible. IOU _{x %} defines the predicted contour as the entire distribution when the cumulative probability of the groundtruth is at least x % but the prediction is less than x %, resulting in the presence of many small values to inflate the union and decrease the IOU score. However, we observe that this is rare in practice and does not have a significant impact on the statistics.

5.1.2 Residual representations

In Table 2, we find that both forms of residual representations results in a decreased mass error. For \mathcal{L}_1 loss, regressing the residual map directly as opposed to adding the input density map to the output also provides the additional benefit of reducing the variance between runs across all metrics.

While the function to be optimised is equivalent for either residual representation, the learning criteria is fundamentally different, and we hypothesise that regressing the residual map provides a more effective criteria due to focusing on the values that change the most across timesteps. This has the effect of the gradient providing information that is explicit for optimising the function that outputs the residual. Otherwise, when regressing the non-residual map,

	\mathcal{L}_2 (1e-9)		IOU ₅₀ (%)		Mass error (1e-2)	
	2018	2016	2018	2016	2018	2016
Base-12	2.49 _{0.08}	2.52 _{0.06}	86.68 _{0.22}	86.75 _{0.17}	1.63 _{0.05}	1.59 _{0.03}
\mathcal{L}^R	2.49 _{0.01}	2.54 _{0.04}	86.73 _{0.06}	86.75 _{0.03}	1.46 _{0.00}	1.44 _{0.03}
Base-18	2.33 _{0.00}	2.38 _{0.02}	87.19 _{0.06}	87.26 _{0.04}	1.51 _{0.02}	1.47 _{0.01}
\mathcal{L}^R	2.32 _{0.01}	2.40 _{0.01}	87.26 _{0.08}	87.27 _{0.06}	1.37 _{0.01}	1.36 _{0.01}
Base-24	2.20 _{0.07}	2.30 _{0.04}	87.51 _{0.23}	87.48 _{0.17}	1.45 _{0.05}	1.46 _{0.02}
\mathcal{L}^R	2.25 _{0.01}	2.37 _{0.03}	87.41 _{0.07}	87.37 _{0.04}	1.37 _{0.01}	1.40 _{0.02}
Fcast-12	7.86 _{0.36}	57.36 _{0.17}	75.98 _{0.53}	43.20 _{0.14}	3.20 _{0.11}	3.10 _{0.08}
\mathcal{L}^R	8.04 _{0.16}	57.44 _{0.15}	75.74 _{0.24}	43.17 _{0.12}	3.12 _{0.10}	3.06 _{0.06}
Fcast-18	6.55 _{0.12}	57.45 _{0.08}	78.07 _{0.16}	43.34 _{0.04}	2.72 _{0.05}	2.71 _{0.07}
\mathcal{L}^R	6.62 _{0.15}	57.52 _{0.24}	78.01 _{0.23}	43.36 _{0.03}	2.63 _{0.04}	2.64 _{0.03}
Fcast-24	5.84 _{0.13}	57.63 _{0.39}	79.28 _{0.25}	43.38 _{0.16}	2.49 _{0.05}	2.50 _{0.02}
\mathcal{L}^R	5.81 _{0.07}	57.53 _{0.06}	79.40 _{0.14}	43.45 _{0.05}	2.35 _{0.02}	2.42 _{0.00}
SSH-12	11.14 _{1.07}	48.58 _{0.68}	71.34 _{1.43}	46.67 _{0.55}	4.56 _{0.54}	4.53 _{0.51}
\mathcal{L}^R	9.69 _{0.55}	51.90 _{2.54}	73.13 _{0.64}	45.68 _{1.02}	3.81 _{0.25}	3.86 _{0.18}
SSH-18	8.82 _{0.31}	49.73 _{1.17}	74.41 _{0.46}	46.70 _{0.52}	3.66 _{0.16}	3.69 _{0.13}
\mathcal{L}^R	8.16 _{0.26}	51.84 _{2.02}	75.39 _{0.35}	46.01 _{0.88}	3.26 _{0.14}	3.39 _{0.07}
SSH-24	8.03 _{0.29}	49.36 _{1.02}	75.55 _{0.44}	46.99 _{0.47}	3.36 _{0.13}	3.44 _{0.11}
\mathcal{L}^R	7.29 _{0.60}	51.26 _{1.90}	76.66 _{0.92}	46.41 _{0.74}	3.00 _{0.28}	3.19 _{0.19}

Table 3 Comparison of modelling scenarios (i.e. input to the network). *Base* refers to inputting the velocity at both t and $t+1$, *Fcast* at t only, and *SSH* at both t and $t+1$ for the variable sea surface height. Results for models trained using both \mathcal{L}_1 and \mathcal{L}_1^R are presented, and Model- X indicates the number of epochs the model was trained for.

the values that have not changed introduces redundancy and dilutes the training signal. The difference between the two representations is therefore likely to be dependent on the dataset.

Given the statistics in Table 1, the drifts in our dataset appear to have a relatively low degree of redundancy. IOU₅₀ is low between D^t and D^{t+1} , the amount of displaced density is $\sim 70\%$ of the distribution sum, and the MSE between D^t and D^{t+1} is greater than the MSE between D^t and zeros.

Table 2 also presents results for combining the two residual representations, which in practice has the effect of requiring the negative component of the drift to be scaled by a factor of two. While the performance is only marginally impacted, the measurable difference reinforces the importance of defining a logical learning criteria. Table 3 shows how this can be even more significant when learning to model more complex tasks.

5.2 Modelling drift given imperfect knowledge of the flow

We experiment with two scenarios of learning to model drift given imperfect knowledge. The first (denoted *Fcast*) completely removes information of the velocity at $t+1$, thereby requiring the model to implicitly forecast the velocity’s future state. The second replaces information of the velocity field with sea surface height (SSH), a variable which provides significant utility in the field of ocean modelling due to its global availability from satellite observations. It

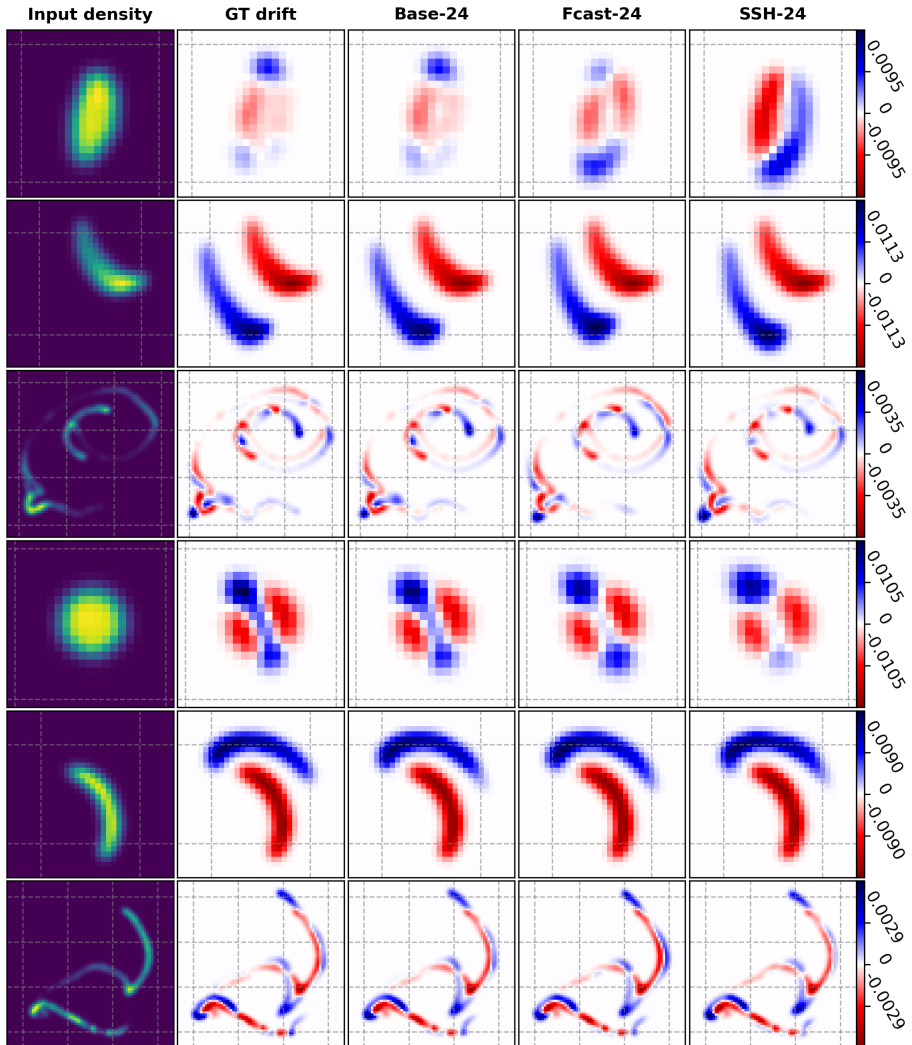


Fig. 3 Sample predictions for the three modelling scenarios trained with \mathcal{L}_1^R . A range of uncertainty levels are shown, where plots are zoomed on relevant foreground regions (grid scale: 20^2 px). Top 3 rows: samples from 2016, bottom 3 rows: samples from 2018. Column 1 shows density values while columns 2-5 show the change of density values between timesteps.

is implicative of sea surface currents at large scales, but does not describe the small scale phenomena that is important for short-term drift modelling. These two scenarios involve learning to recover missing information in the spatial and temporal dimensions, respectively.

We use the exact same architecture and optimisation scheme described in Section 4.2 with the only difference being the input to the network. The forecasting scenario inputs a 3-channel matrix (U_t, V_t, D_t) and the SSH scenario inputs a 3-channel matrix (SSH_t, SSH_{t+1}, D_t) . Table 3 presents results for

both \mathcal{L}_1 and \mathcal{L}_1^R , as well as for models trained over different epoch lengths (12, 18, 24).

We observe that the harder modelling scenarios suffer from a generalisation issue when evaluated on a year that was not seen during training (2016). The error for 2016 appears to have a lower bound which is dependent on the modelling scenario, where it neither decreases nor increases as the 2018 validation error decreases (e.g. when training over more epochs). This suggests that the issue is due to data drift rather than overfitting, where the underlying distribution is changing over time. We can also see that the mass error does not change between years, which informs us that the predicted drifts are still physically plausible. With that said, SSH observes an increased lower bound and an increased mass error for 2016 when using \mathcal{L}_1^R as opposed to \mathcal{L}_1 . This implies that overfitting may still be an issue and that \mathcal{L}_1 may help to provide a regularisation effect. Nevertheless, we hypothesise that performance on unseen years could be improved by broadening the distribution represented in the training set by sampling from more years. A direction for future work would be to evaluate the relationship between the number of years represented and the generalisation error.

Empirical observations from Fig. 3 show that predictions made in the context of the two harder scenarios are generally coherent with respect to large-scale patterns in the groundtruth. However, local details are not always modelled correctly, especially for the unseen year of 2016. For example, columns 1 (2016) and 4 (2018) of Fig. 3 highlights the difficulties of modelling a local divergence in the flow. Occasionally, the divergence is modelled but with an inaccurate weighting of the displacement, and other times the divergence may not be modelled at all. The latter appears to be more common with SSH. While the issue of generalisation is apparent, it is still worth noting that our models have not reached capacity as Table 3 shows a steady improvement when training over more epochs. Therefore, it is possible that the aforementioned issues are not fundamental limitations of the modelling scenarios.

5.3 Validation against artificial flow scenarios

In order to gain insight into the internal biases of our models, we validate them against very simple yet extreme scenarios. These scenarios are static in time, with the velocity magnitude being homogeneous over the field at 0.11 m/s (the mean over the 2018 dataset). Fig. 4 shows 6 example scenarios for our base and forecasting models. We are unable to validate our SSH model due to the inability to explicitly relate velocity to SSH.

In general, we observe that our base model is able to replicate the expected drift very well despite such extreme scenarios likely being absent in the training set. While the forecasting model is able to model the general idea of the expected flow scenario, we observe that local details are incorrect due to its internal bias of what it expects the next state should be. For example, row 2 of Fig. 4 presents a very simple scenario where the expected drift is a linear translation along the horizontal axis. We can infer from its prediction that the

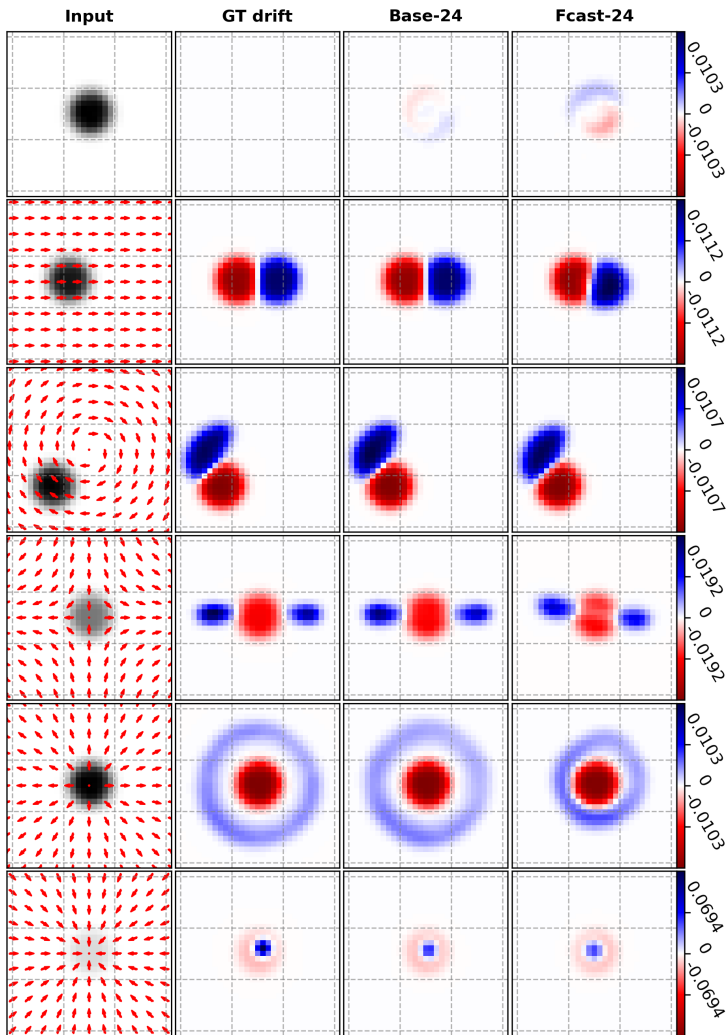


Fig. 4 Predictions made over a range of artificial flow scenarios. From top to bottom: zero, constant, vortex, diverging, repelling, and sink. Column 1 shows the superimposed input velocity field and density map. All input density maps are identical but share the same colour scale with the drift maps in order to highlight the relative change in density.

model expects the velocity field to undergo downwards curvature in the following state. Similar biases can be observed in other examples such as in rows 4 (diverging) and 5 (repelling). While these exact scenarios do not exist in our dataset, the model's inherent pull is likely to be informative of the dataset's average flow evolution in similar contexts. The forecasting model performs well under an artificial vortex scenario likely due to the natural abundance of similar contexts in the dataset.

Row 1 of Fig. 4 presents a unique scenario in which the input density is expected to remain entirely static due to being presented with a velocity field of

zero flow. This scenario only occurs in our dataset for land values, where input density values cannot exist and output density values are excluded from the loss function. We observe that this has the effect of predicting the background as small non-zero values except for the surrounding region of the foreground. The foreground generally appears to remain static, although some degree of drift is observed, with the amount of displacement being 6.6% and 17.5% of the input density’s maximum value for the *Base* and *Fcast* models, respectively. The respective IOU_{50} scores are relatively high at 96.3% and 82.8%.

Row 6 of Fig. 4 shows an extreme example of the density values being compressed to a very dense region. The maximum value increases by a factor of 6.8, but the *Base* and *Fcast* models are only able to model an increase factor of 2.8 and 2.5, respectively, and in the process loses 35% and 28% of the initial mass. As the general idea of compression has been modelled correctly, the loss of mass suggests that the models are sensitive to the absolute density value that can be modelled.

6 Conclusion

We proposed a deep learning framework for modelling Lagrangian drift probabilistically under the influence of uncertainty. Its flexibility allows arbitrary modelling scenarios to be considered, either with respect to the uncertainty distribution being modelled or the underlying flow field being given as input. We demonstrated our framework in the context of modelling floating objects at sea whose initial positions are uncertain. Groundtruth probabilistic drifts were generated by simulating trajectories on surface currents output by an ocean model. We considered three input flow field scenarios for reconstructing the groundtruth drifts: (1) emulator — surface currents at t & $t+1$, (2) forecast — surface currents at t only, and (3) SSH — sea surface height at t & $t+1$. We observe that for the harder scenarios of (2) and (3), the models suffer from an inability to generalise well to out of distribution flow scenarios from a different year. Several toy examples of artificial flows helped to give insight into the biases learned from the training distribution for the forecast scenario. We believe that reducing these biases by improving the representativity of the training distribution should be a key focus for future work.

Acknowledgements

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.

References

- [1] Adrian E. Gill. Atmosphere-ocean dynamics. Academic Press, 30, 1982.
- [2] Erik Van Sebille, Stephen M. Griffies, Ryan Abernathey, Thomas P. Adams, Pavel Berloff, Arne Biastoch, Bruno Blanke, Eric P. Chassignet, Yu Cheng, Colin J. Cotter, et al. Lagrangian ocean analysis: Fundamentals and practices. Ocean Modelling, 121:49–75, 2018.
- [3] Inga M. Koszalka, Thomas W.N. Haine, and Marcello G. Magaldi. Fates and travel times of Denmark Strait overflow water in the Irminger Basin. Journal of Physical Oceanography, 43(12):2611–2628, 2013.
- [4] Andre W Visser. 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] JHt LaCasce. Statistics from lagrangian observations. Progress in Oceanography, 77(1):1–29, 2008.
- [6] Mengjiao Han, Sudhanshu Sane, and Chris R. Johnson. Exploratory Lagrangian-based particle tracing using deep learning. arXiv preprint arXiv:2110.08338, 2021.
- [7] Matthew D. Grossi, Miroslav Kubat, and Tamay M. Özgökmen. Predicting particle trajectories in oceanic flows using artificial neural networks. Ocean Modelling, 156:101707, 2020.
- [8] Yong-Wook Nam, Hwi-Yeon Cho, Do-Youn Kim, Seung-Hyun Moon, and Yong-Hyuk Kim. An improvement on estimated drifter tracking through machine learning and evolutionary search. Applied Sciences, 10(22):8123, 2020.
- [9] Nikolas O. Aksamit, Themistoklis Sapsis, and George Haller. Machine-learning mesoscale and submesoscale surface dynamics from lagrangian ocean drifter trajectories. Journal of Physical Oceanography, 50(5):1179–1196, 2020.
- [10] Jiawei Zhuang, Dmitrii Kochkov, Yohai Bar-Sinai, Michael P. Brenner, and Stephan Hoyer. Learned discretizations for passive scalar advection in a two-dimensional turbulent flow. Physical Review Fluids, 6(6):064605, 2021.
- [11] Yann Ourmières, Bruno Zakardjian, K Béranger, and C Langlais. 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] Madec Gurvan, Romain Bourdallé-Badie, Jérôme Chanut, Emanuela Clementi, Andrew Coward, Christian Ethé, Doroteaciro Iovino, Dan Lea, Claire Lévy, Tomas Lovato, Nicolas Martin, Sébastien Masson, Silvia Mocavero, et al. NEMO ocean engine, 2017.
- [13] Karen Guihou, Julien Marmain, Yann Ourmieres, Anne Molcard, Bruno Zakardjian, and Philippe Forget. 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] Jérémy Mansui, Anne Molcard, and Yann Ourmières. Modelling the transport and accumulation of floating marine debris in the Mediterranean basin. *Marine pollution bulletin*, 91(1):249–257, 2015.
- [15] Philippe Delandmeter and Erik van Sebille. The Parcels v2.0 Lagrangian framework: new field interpolation schemes. *Geoscientific Model Development*, 12(8):3571–3584, 2019.
- [16] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587, 2017.
- [17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *International Conference on Computer Vision*, pages 2961–2969, 2017.
- [18] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention*, pages 234–241, 2015.
- [19] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.
- [20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [21] Soufiane Hayou, Eugenio Clerico, Bobby He, George Deligiannidis, Arnaud Doucet, and Judith Rousseau. Stable resnet. In *International Conference on Artificial Intelligence and Statistics*, pages 1324–1332. PMLR, 2021.
- [22] Marcela Carvalho, Bertrand Le Saux, Pauline Trouvé-Peloux, Andrés Almansa, and Frédéric Champagnat. On regression losses for deep depth estimation. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 2915–2919. IEEE, 2018.
- [23] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101, 2017.
- [24] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983, 2016.
- [25] Jerry Ma and Denis Yarats. On the adequacy of untuned warmup for adaptive optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8828–8836, 2021.
- [26] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. arXiv preprint arXiv:1710.03740, 2017.