Learning Dynamical Systems from Partial Observations

Ibrahim Ayed*
ibrahim.ayed@lip6.fr
Sorbonne Université, UMR 7606, LIP6
Theresis lab, Thales

Emmanuel de Bézenac*
emmanuel.de-bezenac@lip6.fr
Sorbonne Université, UMR 7606, LIP6

Arthur Pajot
Sorbonne Université, UMR 7606, LIP6

Julien Brajard
Sorbonne Université, UMR 7159, LOCEAN
Nansen Environmental and Remote Sensing Center (NERSC)

Patrick Gallinari
Sorbonne Université, UMR 7606, LIP6
Criteo AI Lab

In this work, we consider parametrized evolution equations of the form :

$$\frac{dX_t}{dt} = F_\theta(X_t) \tag{1}$$

Many phenomena studied in physics, computer vision, biology (Mitchell and Schaeffer (2003)), geoscience (Ferguson (1988)), finance (Y. Achdou and Lelievre (2007)), etc... obey a general equation of this form. Here the goal is to learn a θ such that the solution fits measurements. However, there is a major drawback to this approach: for most real-world applications, the variables describing the system are not fully visible to external sensors (Carrassi et al. (2018)), *e.g.* when studying ocean's circulation, variables contained in the system's state such as surface temperature or salinity are observable via satellite imaging, while others subsurface variables are substantially more difficult or costly to observe. In this case, the state is said to be *partially observable*.

We tackle this problem of learning spatio-temporal dynamical systems where the evolution equations are unknown, and only partial observations are available, using neural networks to model their evolution. This approach yields improvements on the forecast of observations, compared against state of the art deep learning based methods for spatio-temporal data, when used to predict observations derived from simulations of the Navier-Stokes equation.

To summarize, our main contributions are the following:

- We propose a general model, parametrized with neural networks, which learns a state representation and its dynamics given partial observations.
- We show that the states produced in the partially observable case by the model aren't necessarily interpretable and propose two settings where we prescribe the hidden dynamics to match the canonical states[2].
- We present experiments on the Navier-Stokes equations exploring the properties of our model in each setting.

*Equal contribution

[2]Which we define here as the representation used by practitioners, generally formed with physically meaningful variables.

Second Workshop on Machine Learning and the Physical Sciences (NeurIPS 2019), Vancouver, Canada.

1 General Approach

1.1 General model

A natural optimization problem would be the following:

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & \mathbb{E}_{Y \in \text{Dataset}} \left[\mathcal{J}(Y, \mathcal{H}(X)) \right] \\ \text{subject to} \quad & \frac{\mathrm{d}X_t}{\mathrm{d}t} = F_\theta(X_t), \\ & X_0 = g_\theta(Y_0^{-k}) \end{aligned} \tag{2}$$

We take:

$$\mathcal{J}(Y, \widetilde{Y}) = \sum_{t=0}^{T} \|Y_t - \widetilde{Y}_t\|^2 \tag{3}$$

The dataset is a set of the form $\{(Y_{-k+1}^{(i)}, ..., Y_0^{(i)}, ..., Y_T^{(i)})\}$, where all observations $Y^{(i)}$ are supposed to be generated through the same underlying dynamical system, with different initial conditions. $\mathcal{H}$ models the loss of information between the state X defining the dynamics of the studied system and the observations Y which can be cheaply and directly measured[3].

1.2 Learning An Ill-Posed Problem

We now consider the more specific situation where we take Y and X to be vector-valued spatio-temporal fields with values respectively in $\mathbb{R}^l$ and $\mathbb{R}^d$ where $l \leq d$. This last operator is taken as a linear projection. Without loss of generality, we can thus consider Y to be constituted by the first l components of X and we have the result:

Proposition 1. *If[4] $l < d$ and the parametric families are universal approximators, for any ϵ, the optimization problem equation 2 admits an infinite number of solutions with a loss lower than ϵ arbitrarily close from non-canonical state representations.*

This result shows in particular that solving the optimization problem defining our model doesn't necessarily give a state representation corresponding to the desired canonical one. In particular, while observations may still be accurately forecast, if the problem is correctly solved, there can be a problem with the interpretation of the other hidden components of the state. In the following, we try to mitigate this issue by considering two possible settings.

1.3 Setting 1: Jointly Trained (JT) States

In this setting, we choose to fix the architectures of g_θ and F_θ and optimize without additional information. The dataset used here is thus only composed of observations and is of the form $\{(Y_{-k+1}^{(i)}, ..., Y_0^{(i)}, ..., Y_T^{(i)})\}$. In the following, the states learned in this setting will be referred to as **Jointly Trained** states.

The following proposition tells us that the non-structured JT states can be interpreted:

Proposition 2. *There exists an invertible function g which transforms jointly learned states into canonical states.*

1.4 Setting 2: Feeding in a Canonical Initial Condition

A weak way to impose some structure over the learnt states is to remove g and prescribe an initial state with canonical structure[5] Thus, in this setting, the dataset used here is of the form

[3] In practice, this can for example be the case in satellite imaging between surface measurements and inaccessible in-depth variables. In other cases, some variables are simply more difficult to measure and therefore can't be available in sufficient quantities for training.

[4] The hypothesis made here correspond to those of our experiments but it is possible to obtain a more general version at the cost of a lengthier proof.

[5] This comes at a cost: The algorithm now has to take a full state as input for each sequence of observations.

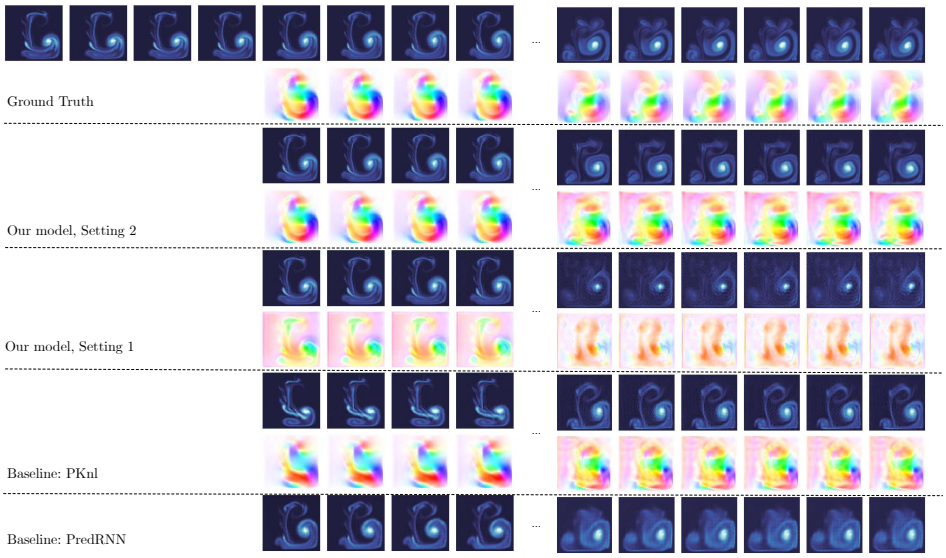

Figure 1: Forecasting the Navier Stokes equations 30 time-steps ahead with different models, starting from a given initial condition. More samples from our model are given in section C of the appendix.

$\{X_0^{(i)}, Y_1^{(i)}, ..., Y_T^{(i)})\}$ and the number of needed states is T **times less** than the number of observations.

There still are infinitely many possible state representations which produce accurate forecasts for observations, even when X_0 is fed as an input to the model. However, by correctly parametrising F, we can hope to conserve the structure of X_0 throughout the forecasts.

2 Experiments

In this section, we present experiments conducted on simulations of the Navier-Stokes equation. Note that we have also evaluated our model on state-of-the-art simulations of ocean circulation. For clarity, the results have been deferred to the appendix, Section B.

2.1 Forecasting Observations

Table 1: Relative MSE $\frac{1}{T}\frac{1}{|\Omega|}\sum_{k=1}^{T}\sum_{x\in\Omega}\frac{\|\mathcal{H}(X_k(x))-Y_k(x)\|_2}{\|Y_k(x)\|_2}$ for our model and two different baselines, at different temporal horizons on the Navier Stokes equations.

Model	$T=5$	$T=10$	$T=50$
Ours	0.152	0.243	0.650
PKnI de Bézenac, Pajot, and Gallinari (2018)	0.194	0.221	0.752
PRNN (Wang et al. (2018))	0.170	0.227	0.719

Figure 1 shows a sample of the predictions of our system over the test set for the Navier Stokes equations. The good results it shows are confirmed by table 1. Our model is able to predict observations up to a long forecasting horizon, which means that it has managed to learn the dynamical system. Note that the initial states used at test time have never been seen at training time which means that the optimization problem was solved correctly without over-fitting. The cost function and the supervision were only defined at the level of observations in accordance with our setting. An interesting remark is to observe that the jointly trained model is slightly less accurate than the one given X_0, which makes sense as this last algorithm is given a few additional full states when JT isn't given any. Samples for long term forecasts of our model can be seen in the appendix for the Navier Stokes equations.

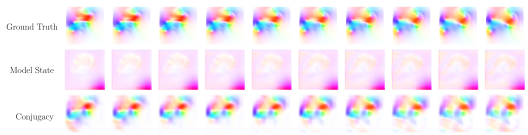

Figure 2: Example of a sequence of hidden states transformed by the calculated conjugacy.

2.2 Restructuring Jointly Trained States

Proposition 2 shows that there must exist a way to transform JT states into canonical ones, which would make them more palatable and easier to interpret. In order to confirm this theoretical result empirically, we solved the regression problem between JT and canonical states[6].

Figure 2 shows an example of the output from the conjugacy yielded by this transformation we learned: It allows us to transform the non-structured hidden states of the jointly trained model into interpretable states corresponding to the canonical representation. From a quantitative point of view, after 5 predictions, the average cosine similarity over the whole test set goes from 0.192 in the jointly trained representation to 0.582 when transformed. While this result is far from perfect[7], it still shows promise and demonstrates that this approach could be applied in many cases.

2.3 Imposing the Initial Condition Prescribes the Hidden Dynamics

Figure 1 shows that in **setting 2**, when we add a full initial state, our model is able to forecast not only observations but also the dynamics of the hidden components of the state. This is a surprising result: Even though this model gets additional structured information at the input, there are still an infinite number of ways to transport that information through time-steps and to store it into the state representation.

Table 2: Cosine similarity $\frac{1}{T}\sum_{k=1}^{T}\frac{1}{|\Omega|}\sum_{x\in\Omega}\frac{\langle u(x),v(x)\rangle}{\|u(x)\|\|v(x)\|}$ scores for our models and another baseline, at different temporal horizons on the Navier Stokes equations.

MODEL	$T=5$	$T=10$	$T=50$
OURS (WITH INITIAL STATE AS INPUT)	0.798	0.679	0.483
PKNI	0.243	0.207	0.098

Table 3: Ablation study for our model, at different temporal horizons on the Navier Stokes equations

MODEL	T=5		T=10		T=50	
	MSE	COSINE	MSE	COSINE	MSE	COSINE
OURS	0.118	0.798	0.180	0.679	0.628	0.483
RESNET	0.288	0.604	0.391	0.333	0.73	0.032
UNET	0.659	0.069	0.692	0.028	0.84	0.023
RESNET NO SKIP	0.615	0.162	0.71	0.060	0.897	-0.04

Table 3 shows an ablation study conducted on our model with other possible parametrizations, all with roughly the same number of parameters as our model:

- At least in the case of the Navier-Stokes equations, our model, with a simple solver for an equation parametrized through a residual network, allows to forecast unsupervisedly the dynamics of the hidden dynamics of the state;
- The fact that a solver is used, instead of a direct regression model, appears to be very important, as comparisons to other standard powerful architectures show.

However, this still doesn't explain why this works for the hidden components, as the problem is ill-posed nonetheless. We hypothesize that the architecture of the network used to parametrize the

[6]With a small number of canonical states being provided.

[7]In particular, the choice of the regression algorithm isn't obvious and the size of the needed dataset will depend on this choice as well as on the desired accuracy, as with any regression problem.

equation is biased towards preservation of the input code, which happens to be that of the canonical state because X_0 is fed into it[8]

3 Conclusion

We present in this paper a general data-driven model for space-time processes when the state is only partially observable. We show that partial observability introduces ill-posedness in the determination of an interpretable state representation then propose two methods to solve this issue: This allows to demonstrate that non-structured states can be interpreted when correctly transformed and that the model, when fed with a structured interpretable state with a well parametrized evolution term, can forecast unsupervisedly the hidden dynamics of the state. The theoretical analysis is confirmed through experiments on raw simulations of the Navier-Stokes equations and comparisons with two competitive baselines.

[8] A similar kind of phenomenon is also empirically observed in the unsupervised domain translation field with the success of the CycleGAN model which is explored from this point of view in de Bézenac, Ayed, and Gallinari (2019).

References

Carrassi, A.; Bocquet, M.; Bertino, L.; and Evensen, G. 2018. Data assimilation in the geosciences: An overview of methods, issues, and perspectives. *Wiley Interdisciplinary Reviews: Climate Change* 9(5):e535.

de Bézenac, E.; Ayed, I.; and Gallinari, P. 2019. Optimal unsupervised domain translation. *CoRR* abs/1906.01292.

de Bézenac, E.; Pajot, A.; and Gallinari, P. 2018. Deep learning for physical processes: Incorporating prior scientific knowledge. In *ICLR*.

Ferguson, J. 1988. Geological applications of differential equations. In Springer., ed., *Mathematics in Geology*. chapter 8, 216–237.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 770–778.

Madec, G. 2008. *NEMO ocean engine*. Note du Pôle de modélisation, Institut Pierre-Simon Laplace (IPSL), France, No 27, ISSN No 1288-1619.

Mitchell, C. C., and Schaeffer, D. G. 2003. A two-current model for the dynamics of cardiac membrane. *Bulletin of mathematical biology* 65(5):767–793.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in pytorch.

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-net: Convolutional networks for biomedical image segmentation. *CoRR* abs/1505.04597.

Wang, Y.; Gao, Z.; Long, M.; Wang, J.; and Yu, P. S. 2018. Predrnn++: Towards a resolution of the deep-in-time dilemma in spatiotemporal predictive learning.

Y. Achdou, O. B., and Lelievre, T. 2007. Partial differential equations in finance.

A Practical details

A.1 Training and Inference Algorithms

The formulation above closely ressembles control problems where a given controllable dynamical system is constrained to optimize a certain objective, our aim here is different as our goal is to find the dynamical system fitting a certain set of constraints, here provided through the observations Y. In practice, the optimization problem defined here can be solved using gradient descent methods. There are many methods to calculate the gradient, we used here an explicit Euler solver which allowed us to use backpropagation in order to compute the gradient. F_θ and g_θ can be parameterized as a neural network or by another parametric family, the only constraint being that it is differentiable almost everywhere with respect to θ, while we also have to suppose that $\mathcal{H}$ is differentiable.

A natural *training* algorithm would then be algorithm 1.

Algorithm 1 Training Procedure

Input: Training samples $\{(Y^{-k}), Y_{+l}\}$.
Guess initial parameters θ
while not converged **do**
  Randomly select sample sequence $\{(Y^{-k}), Y^{+l}\}$
  $X_0 \leftarrow g_\theta(Y_{-k})$
  Solve Forward $\frac{dX_t}{dt} = F_\theta(X_t),\ X(0) = X_0,\ t \in [0, l]$
  Solve Backward $\dfrac{d\lambda_t}{dt} = A_t\lambda_t + B_t,\ \lambda_l = 0,\ t \in [0, l]$
  Compute gradient $\frac{\partial \mathcal{J}}{\partial \theta}(X^\theta)$
  Update θ in the steepest descent direction
end while
Output: Learned parameters θ.

For *inference*, the parameters being learnt and fixed, we simply calculate $X_0 = g_\theta(Y^{-k})$ then use it as an initial condition to solve the equation parametrized by F_θ which gives us X_t for any t.

A.2 The Dataset

For those experiments, we have worked with the incompressible Navier-Stokes equations. We have taken the observations to be the density of the fluid while the hidden components are the two-dimensional velocity field.

We have produced 600 separate simulations with **independently generated initial conditions**, the simulations have then been subsampled in time to produce a time-step of $2.5s$ and a total length of 50 time-steps per simulation. We have taken 300 from those simulations to construct the training set, 200 for validation and 100 for test. In particular, this means that the sequences used in the test results we present and analyze below are produced by **initial conditions the model has never seen**. For both settings, this gives us a total of 15000 observations for the training set and 10000 for the test set. In setting 1, for the restructuring of JT states experiment, we used 500 additional full states to train the transformation. In setting 2, we use an additional 2500 full states for training and 1666 for testing.

As stated before, one also has to choose a training horizon T, to construct the used dataset of the form $\{(Y^{(i)}_{-k+1}, ..., Y^{(i)}_0, ..., Y^{(i)}_T)\}$ for **setting 1** and $\{X^{(i)}_0, Y^{(i)}_1, ..., Y^{(i)}_T)\}$ for **setting 2**. We have treated T as a hyperparameter of the model and have chosen it to be equal to 6. An important observation is that the higher T, the more memory demanding the training will be and the more carefully the gradient descent has to be done, especially at the first steps (by tuning the learning rate, scheduled sampling,...). However, we have observed that models with higher horizons tend to generalize better and forecast more accurately for farther time horizons, which makes sense as it makes the model take into account long term effects.

An other important misconception to avoid is to confuse the training horizon T with the inference horizons at test time: For example, a model which is trained for sequences with $T = 6$ can be very accurate for longer time horizons as we show in the results below.

A.3 Implementation

In practice, the cost functional $\mathcal{J}$ is estimated on a minibatch of sequences from the dataset and optimized using stochastic gradient descent. Throughout *all* the experiments, F_θ is a standard residual network He et al. (2016), with 2 downsampling layers, 6 residual blocks, and bilinear up-convolutions instead of transposed convolutions. In the experiments for setting 1, we parametrize g_θ as a Unet Ronneberger, Fischer, and Brox (2015). To discretize the forward equation equation **??** in time, we use a simple Euler scheme. Note that the discretization step-size may differ from the time interval between consecutive observations; in our case, we apply 3 Euler steps between two observations, *i.e.* $\delta t = \frac{1}{3}$. For the spatial discretization, we use the standard grid discretization induced by the dataset. The weights of the residual network θ are initialized using an orthogonal initialization. Our model is trained using a exponential scheduled sampling scheme with exponential decay, using the Adam optimizer, with a learning rate set to 1×10^{-5}. We use the Pytorch deep learning library Paszke et al. (2017).

A.4 Baselines and Metrics

We compare our models to two different baselines:

PKnI This is a physics-informed deep learning model de Bézenac, Pajot, and Gallinari (2018), where prior physical knowledge is integrated: it uses an advection-diffusion equation to link the velocity with the observed temperatures, and uses a neural network to estimate the velocities.

PRNN Wang et al. (2018) This is a heavy-weight, state of the art model used for video prediction tasks. It is based on a Spatiotemporal Convolutional LSTM that models spatial deformations and temporal variations simultaneously.

We use a renormalized relative squared error as a metric for observations:

$$\frac{1}{K}\frac{1}{|\Omega|}\sum_{k=1}^{K}\sum_{x\in\Omega}\frac{\|\mathcal{H}(X_k(x)) - Y_k(x)\|_2}{\|Y_k(x)\|_2} \tag{4}$$

To evaluate the quality of the hidden states, we use cosine similarity between the model's hidden state and the true hidden state of the system[9]:

$$\frac{1}{K}\sum_{k=1}^{K}\frac{1}{|\Omega|}\sum_{x\in\Omega}\frac{\langle u(x), v(x)\rangle}{\|u(x)\|\,\|v(x)\|} \tag{5}$$

For the velocity vector field representation, color represents the angle, and the intensity the magnitude of the associated vectors.

B Taking our model to the limit: Forecasting ocean circulation dynamics

In this section, we use our model to study Sea Surface Temperatures dynamics as modeled by the Glorys2v4 simulations. We quickly describe this complex realistic state-of-the-art simulation of ocean circulation then give the declination of our model we used to learn it and present the results as compared to standard baselines.

B.1 Glorys2v4

The Glorys2v4 product is a reanalysis of the global Ocean (and the Sea Ice, not considered in this work). The numerical ocean model is NEMOv3.1 Madec (2008) constrained by partial real observations of Temperature, Salinity and Sea Level. Oceanic output variables of this product are daily means of Temperature, Salinity, Currents, Sea Surface Height at a resolution of 1/4 degree horizontal resolution.

[9]The cosine similarity is relevant for the comparison with PKnI: the norm of its hidden state may not correspond to the ground truth norm.

The NEMO model describes the ocean by the primitive equations (Navier-Stokes equations together with an equation of states).

Let $(\mathbf{i}, \mathbf{j}, \mathbf{k})$ the 3D basis vectors, U the vector velocity, $\mathbf{U} = \mathbf{U}_h + w\mathbf{k}$ (the subscript h denotes the local horizontal vector, *i.e.* over the $(\mathbf{i}, \mathbf{j})$ plane), T the potential temperature, S the salinity, ρ the *in situ* density. The vector invariant form of the primitive equations in the $(\mathbf{i}, \mathbf{j}, \mathbf{k})$ vector system provides the following six equations (namely the momentum balance, the hydrostatic equilibrium, the incompressibility equation, the heat and salt conservation equations and an equation of state):

$$\frac{\partial \mathbf{U}_h}{\partial t} = -\left[(\mathbf{U}.\nabla)\mathbf{U}\right]_h - f\mathbf{k} \times \mathbf{U}_h - \frac{1}{\rho_0}\nabla_h p + D^{\mathbf{U}} + F^{\mathbf{U}}$$
$$\frac{\partial p}{\partial z} = -\rho g$$
$$\nabla.\mathbf{U} = 0$$
$$\frac{\partial T}{\partial t} = -\nabla.(T\mathbf{U}) + D^T + F^T$$
$$\frac{\partial S}{\partial t} = -\nabla.(S\mathbf{U}) + D^S + F^S$$
$$\rho = \rho(T, S, p)$$

where ρ is the *in situ* density given by the equation of the state B.1, ρ_0 is a reference density, p the pressure, $f = 2\Omega.\mathbf{k}$ is the Coriolis acceleration. D^U, D^T and D^S are the parameterizations of small-scale physics for momentum, temperature and salinity, and F^U, F^T and F^S surface forcing terms.

As in subsection **??**, the divergence-free constraint over can be enforced through the Leray operator. Moreover, ρ is a function of other state variables so that the state can be written as:

$$X = \begin{pmatrix} U \\ p \\ S \\ T \end{pmatrix} \text{ and } \mathcal{H}(X) = \overline{T}.$$

where $\overline{T}$ is the daily mean temperature derived from the instantaneous potential temperature T in the model.

B.2 Modeling and Results

This dataset is much more challenging and represents a leap from the fully simulated ones presented before. One reason is obviously the high dimensionality of the system and the absence of a full state as initial input to our system as we only have a proxy over the velocity field. A second one is the fact that we only work over sequences from the same ocean zone while the model functions within a larger area. This makes the dynamics for a single zone **non-stationary** as boundary conditions are constantly shifting, thus violating an important assumption of our method and making it almost impossible to make long term forecasts with a reasonable number of observations. All we can hope for is for the dynamics to be locally stationary so that the model can work well for a few steps.

In order to take into account the observations made above regarding this system, especially the fact that the initial temperatures T_0 (in this case, since the we observe the temperatures, $Y_0 = T_0$) and the proxy of the velocity field $\tilde{w}_0$ provided as initial input is insufficient to represent the full state, we take g_θ in equation equation 2 to be:

$$g_\theta = E_\theta(Y^{(-L)}, \tilde{w}_0) + \begin{pmatrix} Y_0 \\ \breve{w}_0 \\ 0 \end{pmatrix} \tag{6}$$

where E_θ is an encoder neural network. Using E_θ allows us to encode available information from the observations $Y^{(-L)}$ which is not contained in $\breve{w}_0$ nor in T_0. For E_θ, we use the UNet architecture Ronneberger, Fischer, and Brox (2015). This variant, noted **Ours, with Estimation**,

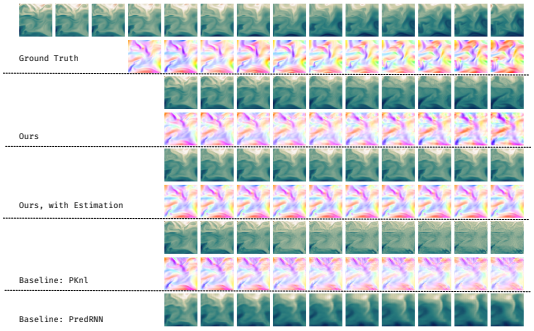

Figure 3: Forecasting Sea Surface Temperatures 10 time-steps ahead with different models, starting from a given initial condition.

shows the potential of our method to be used in settings of varying difficulties and improves on our model, noted **Ours**, in forecasting the hidden components of the state.

Figure 3 and table 4 show that our models are accurate both for observations and hidden states. However, there is a clear drop in performance when compared to the experiments in the Navier-Stokes dataset, even at horizon 10. Still, those experiments show that our model is quite robust even in non-stationary settings.

Table 4: Relative MSE and cosine similarity scores for our models and different baselines, at different temporal horizons on the Glorys2v4 dataset

MODEL	H=5		H=10	
	MSE	COSINE	MSE	COSINE
OURS	0.306	0.671	0.402	0.589
OURS, EST.	0.364	0.718	0.490	0.670
PKNI	0.411	0.448	0.494	0.368
PRNN	0.423	XX	0.546	XX

C Additional Forecasts

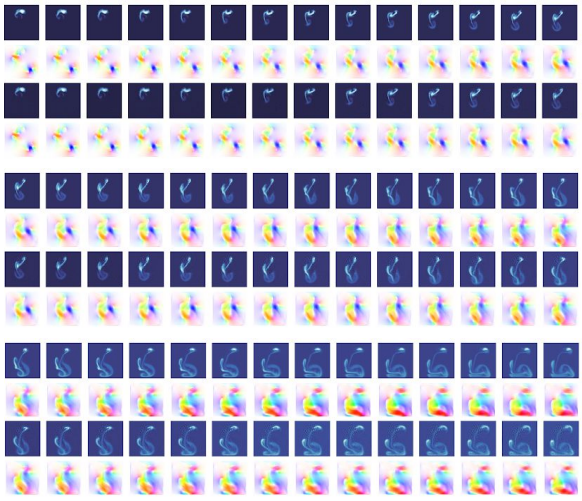

Figure 4: Forecasting the Navier Stokes equations, starting from a given initial condition (not shown here). We forecast 42 time-steps ahead (rows $0, 1 (\mathrm{mod}\ 4)$) and compare results with the ground truth simulation (rows $2, 3 (\mathrm{mod}\ 4)$).

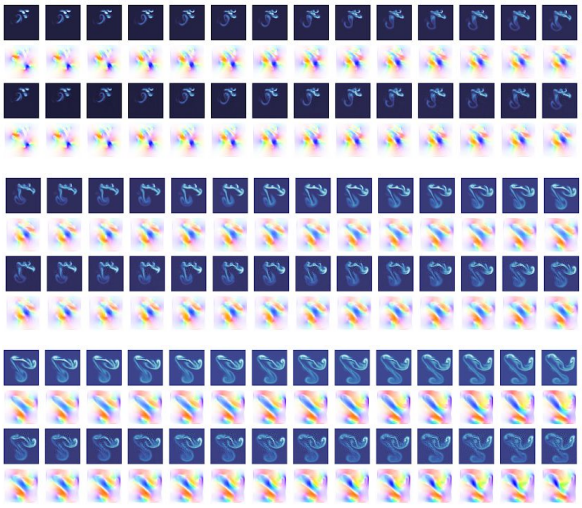

Figure 5: Forecasting the Navier Stokes equations, starting from a given initial condition (not shown here). We forecast 42 time-steps ahead (rows $0, 1 (\mathrm{mod}\ 4)$) and compare results with the ground truth simulation (rows $2, 3 (\mathrm{mod}\ 4)$).

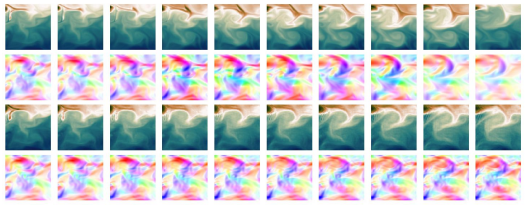

Figure 6: Forecasting Glorys2v4 10 time-steps ahead, starting from a given initial condition (not shown here). Top two rows: ground truth, bottom two rows: model forecasts.

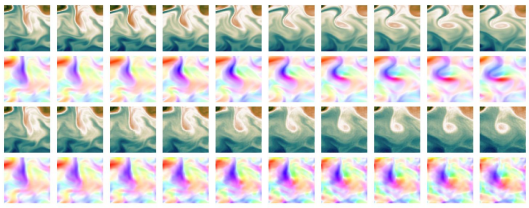

Figure 7: Forecasting Glorys2v4 10 time-steps ahead, starting from a given initial condition (not shown here). Top two rows: ground truth, bottom two rows: model forecasts.

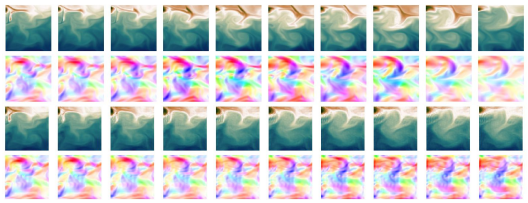

Figure 8: Forecasting Glorys2v4 10 time-steps ahead with estimation step, starting from a given initial condition (not shown here). Top two rows: ground truth, bottom two rows: model forecasts.

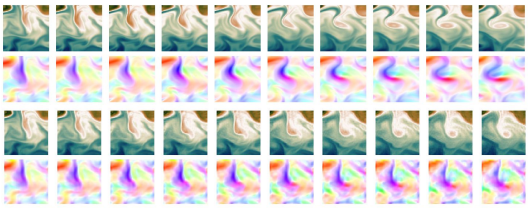

Figure 9: Forecasting Glorys2v4 10 time-steps ahead with estimation step, starting from a given initial condition (not shown here). Top two rows: ground truth, bottom two rows: model forecasts.