Incorporating physical knowledge into deep neural network

Arthur Pajot

▶ **To cite this version:**

Arthur Pajot. Incorporating physical knowledge into deep neural network. Neural and Evolutionary Computing [cs.NE]. Sorbonne Université, 2019. English. NNT : 2019SORUS290 . tel-03015931

HAL Id: tel-03015931
https://theses.hal.science/tel-03015931

Submitted on 20 Nov 2020

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.

THÈSE DE DOCTORAT DE
SORBONNE UNIVERSITÉ

Spécialité
Informatique
École Doctorale Informatique, Télécommunications et Électronique (Paris)

Présentée par

Arthur Pajot

Pour obtenir le grade de
DOCTEUR de SORBONNE UNIVERSITÉ

Sujet de la thèse :

Incorporating Physical Knowledge Into Deep Neural Network

devant le jury composé de :

M. Etienne Mémin	INRIA Renne – IRISA	Rapporteur
M. François Fleuret	IDIAP – EPFL	Rapporteur
M. Gérard Biau	Sorbonne Université – LPSM	Examinateur
Mme. Eniko Székely	Swiss Data Science Center	Examinatrice
M. Nicolas Thome	CNAM – CEDRIC	Examinateur
M. Patrick Gallinari	Sorbonne Université – LIP6	Directeur de thèse

Arthur Pajot: *Incorporating Physical Knowledge Into Deep Neural Network,* © 2019

ABSTRACT

A physical process is a sustained phenomenon marked by gradual changes through a series of states occurring in the physical world. Physicists and environmental scientists attempt to model these processes in a principled way through analytic descriptions of the scientist's prior knowledge of the underlying processes. Despite the undeniable Deep Learning success, a fully data-driven approach is not yet ready to challenge the classical approach for modeling dynamical systems.

We will try to demonstrate in this thesis that knowledge and techniques accumulated for modeling dynamical systems processes in well-developed fields such as maths or physics could be useful as a guideline to design efficient learning systems and conversely, that the ML paradigm could open new directions for modeling such complex phenomena.

We describe three tasks that are relevant to the study and modeling of Deep Learning and Dynamical System : Forecasting, hidden state discovery and unsupervised signal recovery.

RÉSUMÉ

Un processus physique est un phénomène marqué par des changements graduels à travers une série d'états successifs se produisant dans le monde physique. Les physiciens et les climatologues tentent de modéliser ces processus d'une manière fondée sur le principe de descriptions analytiques des connaissances *a priori* des processus sous-jacents. Malgré le succès indéniable de l'apprentissage profond, une approche entièrement axée sur les données n'est pas non plus encore prête à remettre en question l'approche classique de modélisation des systèmes dynamiques.

Nous tenterons de démontrer dans cette thèse que les connaissances et les techniques accumulées pour modéliser des processus de systèmes dynamiques dans des domaines bien développés comme les mathématiques ou la physique, pourraient servir de guide pour concevoir des systèmes d'apprentissage automatique efficaces et, inversement, que l'apprentissage machine pourrait ouvrir de nouvelles directions pour la modélisation de phénomènes très complexes.

Nous décrivons trois tâches pertinentes à l'étude et à la modélisation du lien entre l'apprentissage profond et les systèmes dynamiques : la prévision, la découverte d'états cachés et la reconstruction de signal non supervisé.

ACKNOWLEDGMENTS

Il m'est difficile, voir presque impossible, de me souvenir et d'inclure tous ceux qui ont pu m'aider m'assister ou me soutenir durant ces trois années. Ma thèse a probablement été un des moments clés de ma vie,

Je remercie chaleureusement toutes les personnes qui m'ont aidé pendant l'élaboration de ma thèse et notamment mon directeur Patrick Gallinari, pour son intérêt et son soutien, sa grande disponibilité et ses nombreux conseils durant la rédaction et mon parcours. Une pensée particulière a Emmanuel, Ibrahim et Yuan pour leur aide et collaboration.

Ce travail n'aurait pu être mené à bien sans la disponibilité et l'accueil chaleureux que m'ont témoignés toute l'équipe du MLIA, les anciens comme les nouveaux. Merci au permanent, aux thésards, particulièrement aux gens de mon bureau, pour le soutien, l'écoute dans le travail et en dehors. Merci également a Nadine et Christophe, pour leur disponibilité, leur aide, matériel, administratif et amical. Merci aussi à Julien, Dominique, Anastase, pour leur précieuse discussion.

Au terme de ce parcours, je remercie enfin celles et ceux qui me sont chers. Leurs attentions et encouragements m'ont accompagnée tout au long de ces années. Je suis redevable à mes parents, pour leur soutien moral et matériel et leur confiance indéfectible dans mes choix. Merci également à mes frères et mes soeurs, qui ont assisté à tous mes devoirs. Merci aussi tout particulièrement à mon grand-père, qui a bien voulu relire avec bienveillance mon manuscrit. Merci aussi a ma grand-mère, mes oncles, mes tantes et a tous mes cousin.e.s.

Je remercie aussi beaucoup tous mes amis, Art, Paul, Justine, Léa, Ariane, Robin, Emma, Victor, et tous les autres qui sont trop nombreux pour les citer ici.

Ces remerciements ne serait évidement pas complet, sans les adresser a Sophia, qui a été la pour tout, et m'a aidé pour tout.

Ce travail n'aurait pas été possible sans le soutien de Sorbonne Université et de l'ENS de Rennes, qui m'a permis, grâce à une bourse de recherches et d'enseignement, de me consacrer sereinement à l'élaboration de ma thèse.

J'aimerais dédier cette thèse à ma grand-mère, qui me manque énormément.

CONTENTS

ABSTRACT iii

RÉSUMÉ v

ACKNOWLEDGMENTS vii

Contents ix

LIST OF FIGURES xiii

LIST OF TABLES xvii

ACRONYMS xix

1 INTRODUCTION 1

1.1 Context and Motivation . . . 1

1.2 Tasks . . . 3

1.2.1 Forecasting . . . 3

1.2.2 Hidden State Discovery . . . 4

1.2.3 Inverse Problem . . . 4

1.3 Outline of the thesis . . . 6

2 RELATED WORKS 7

2.1 Forecasting . . . 7

2.1.1 Time Series Forecasting . . . 8

2.1.2 Recurrent Neural Networks . . . 9

2.1.3 Video Prediction . . . 10

2.2 Modeling Differential Equations . . . 13

2.2.1 Ordinary Differential Equations . . . 13

2.2.2 Partial Differential Equations . . . 16

2.2.3 PDE and Neural Networks . . . 18

2.3 Inverse Problem . . . 18

2.3.1 GAN . . . 18

2.3.2 Image Translation . . . 21

2.3.3 Inverse Problem . . . 23

3 INCORPORATING PRIOR KNOWLEDGE IN FORECASTING SPATIO-TEMPORAL DATA 27

3.1 Introduction . . . 28

3.2 Physical Motivation . . . 30

3.2.1 Incorporating the Advection Diffusion Equation . . . 30

3.3 Model . . . 31

3.3.1 Model Description . . . 31

3.3.2 Warping Scheme . . . 33

3.3.3 Loss Function . . . 34

3.4 Experiments . . . 35

3.4.1 Dataset . . . 35

3.4.2 Experimental Setting . . . 36
3.4.3 Model Architecture . . . 37
3.4.4 Baselines . . . 37
3.4.5 Results. . . . 38
3.4.6 Quantitative Results . . . 38
3.4.7 On the Generalization in Space and Time . . . 41
3.5 Conclusion . . . 43
3.5.1 Additional Samples . . . 43
4 LEARNING DYNAMICAL SYSTEMS FROM PARTIAL OBSERVATIONS 47
4.1 Introduction . . . 48
4.2 Methodology . . . 49
4.2.1 Our Approach . . . 49
4.2.2 Models . . . 53
4.3 Experiments . . . 53
4.3.1 Datasets. . . . 54
4.3.2 Experimental setting. . . . 54
4.3.3 Results. . . . 56
4.4 Conclusion . . . 61
5 UNSUPERVISED IMAGE RECONSTRUCTION 65
5.1 Introduction . . . 66
5.2 Models . . . 68
5.2.1 Problem setting. . . . 68
5.2.2 Approach . . . 69
5.3 Method . . . 70
5.3.1 Handling the Likelihood Term . . . 70
5.3.2 Handling the Prior Term . . . 72
5.3.3 Putting everything together . . . 73
5.4 Stochastic Variation . . . 75
5.4.1 Setting . . . 76
5.4.2 Model . . . 77
5.5 Temporal Variation . . . 82
5.5.1 Setting . . . 82
5.5.2 Model . . . 83
5.5.3 Prior handling . . . 83
5.5.4 Likelihood Handling . . . 84
5.6 Experiments . . . 84
5.6.1 Model architectures and Datasets . . . 84
5.6.2 Baselines . . . 90
5.6.3 Deterministic Results . . . 94
5.6.4 Stochastic Results . . . 95
5.6.5 Temporal Results . . . 100
5.6.6 Comparison with Baselines . . . 102

5.6.7 Ablation Study 104
5.7 Conclusion 106
5.8 Additional Samples 107
6 CONCLUSION 115
6.1 Summary of Contributions 115
6.2 Future Direction 117
7 ANNEX 119
7.1 Dataset of Section 4.3.1 119
7.2 Proof of Theorem Theorem 3.1 121
7.3 Proof of Theorem 4.1 124
BIBLIOGRAPHY 127

LIST OF FIGURES

Chapter 1: introduction 1

Chapter 2: related works 7

Figure 2.1 Example of a time series 8

Figure 2.2 A schematic LSTM system 11

Figure 2.3 ODE Net 15

Figure 2.4 GAN Framework 19

Chapter 3: incorporating prior knowledge in forecasting spatio-temporal data 27

Figure 3.1 Estimation Model 32

Figure 3.2 Warping Scheme 33

Figure 3.3 SST Dataset 36

Figure 3.4 Model Architecture 37

Figure 3.5 SST Results 39

Figure 3.6 Flow Direction 41

Figure 3.7 Evaluation of our model's accuracy in time on data from 2006 to 2010 using data from 2011 to 2017 for training. Regions 17 to 20 were used for both periods. Each day, we produce daily forecasts for 6 days ahead and calculate the associated mean square error. The color of the flow represents the direction of the direction each of the vector as illustrated in Figure 3.6. 42

Chapter 4: learning dynamical systems from partial observations 47

Figure 4.1 Setting 1 Model 54

Figure 4.2 Setting 2 Model 55

Figure 4.3 The discretization of our forward equation : a three steps Euler scheme. 55

Figure 4.4 Forecasting the Navier Stokes equations 10 time steps ahead with different models, starting from a given initial condition. 57

Figure 4.5 Forecasting the Navier Stokes equations, starting from a given initial condition (not shown here). We forecast 42 time steps ahead. We show in this figure 3 different sequences of 42 time stted. Top 2 rows correspond to the ground truth and the bottom rows correspond model forecasts. Each sequences is represented as 4 consecutive rows. 59
Figure 4.6 Forecasting the Navier Stokes equations, starting from a given initial condition (not shown here). We forecast 42 time steps ahead. We show in this figure 3 different sequences of 42 time stted. Top 2 rows correspond to the ground truth and the bottom rows correspond model forecasts. Each sequences is represented as 4 consecutive rows. 60
Figure 4.7 Forecasting the shallow water equations, starting from a given initial condition (not shown here). We forecast 42 time steps ahead. We show in this figure 3 different sequences of 42 time steps. Top 2 rows correspond to the ground truth and the bottom rows correspond model forecasts. Each sequence is represented as 4 consecutive rows. 63
Figure 4.8 Forecasting the shallow water equations, starting from a given initial condition (not shown here). We forecast 42 time steps ahead. We show in this figure 3 different sequences of 42 time steps. Top 2 rows correspond to the ground truth and the bottom rows correspond model forecasts. Each sequence is represented as 4 consecutive rows. 64
Figure 4.9 Time interpolations with our approach on the test set. We train our model by regressing to the targets every 3 images (materialized by the red boxes). We then compare the outputs of the model with the unseen ground truth states. 64

Chapter 5: unsupervised image reconstruction 65
Figure 5.1 Likelihood Term 70
Figure 5.2 Prior Term 72
Figure 5.3 Complete Model 74
Figure 5.4 Stochastic Inference 77
Figure 5.5 Stochastic GAN Loss 78
Figure 5.6 Encoding Reconstruction Loss 80
Figure 5.7 Unpaired Variant 91

Figure 5.8 Paired Variant 92
Figure 5.9 CelebA Reconstruction 96
Figure 5.10 CelebA Patch-band Results 96
Figure 5.11 CelebA PixelDark Results 96
Figure 5.12 LSUN Patch-Band Results 97
Figure 5.13 Recipe-1m Remove-Pixel Results 97
Figure 5.14 CelebA Stochastic Patch Results 98
Figure 5.15 CelebA Stochastic DropPixel Results 99
Figure 5.16 SST 100
Figure 5.17 FaceForensics++ 101
Figure 5.18 KTH 101
Figure 5.19 BAIR 102
Figure 5.20 Results with clouds generated at different LWP thresholds. 102
Figure 5.21 Comparison of results with clouds at LWP threshold 70 g/m². *Unable to finish. 102
Figure 5.22 Comparison with supervised baselines for FaceForensics++ with Raindrops. 104
Figure 5.23 Comparison of results for SST data for ablation study . . 104
Figure 5.28 Baseline CelebA Remove Pixel 107
Figure 5.29 Baseline Remove-Pixel-Channel CelebA 107
Figure 5.30 Baseline Convnoise CelebA 108
Figure 5.31 Baseline Patch-Band CelebA 108
Figure 5.32 LSUN Additional Images 109
Figure 5.33 Recipe Additional Images 110
Figure 5.34 LSUN Additional Images 111

LIST OF TABLES

CHAPTER 1: INTRODUCTION 1

CHAPTER 2: RELATED WORKS 7

CHAPTER 3: INCORPORATING PRIOR KNOWLEDGE IN FORECASTING SPATIO-TEMPORAL DATA 27

Table 3.1 Average score and average time on test data. Average score is calculated using the *mean square error* metric (MSE), time is in seconds. 38

Table 3.2 Evaluation of our model's spatial generalization ability. We train our model on two distinct regions and calculate the MSE on both regions for each trained model. 42

CHAPTER 4: LEARNING DYNAMICAL SYSTEMS FROM PARTIAL OBSERVATIONS 47

Table 4.1 Relative MSE and cosine similarity scores for our models and different baselines, at different temporal horizons on the Navier Stokes equations. As the PredRNN does not explicitly model the hidden state, we replace the cosine similarity scores for this baseline with XX. 58

Table 4.2 Relative MSE and cosine similarity scores for our model, at different temporal horizons on the Shallow Water equations 61

Table 4.3 Ablation study for our model, at different temporal horizons on the Navier Stokes equations 61

CHAPTER 5: UNSUPERVISED IMAGE RECONSTRUCTION 65

Table 5.1 Average Mean Squared Error (MSE) on CelebA 95

Table 5.2 MSE, Frechet Inception Distance (FID), and standard deviation on the celebA dataset for three noises. 98

Table 5.3 MSE, FID, and standard deviation on the celebA dataset for three noises and different baselines. 100

Table 5.4 Results for FaceForensics, KTH, and BAIR. Compared with (1) (Alvera Azcarate et al. 2005) and (2) (Newson et al. 2014). *Unable to finish. 103

ACRONYMS

AI	Artificial Intelligence
CNN	Convolutional Neural Network
STN	Spatial Transformer Network
RNN	Recurrent Neural Network
ConvLSTM	Convolutional Long-Short Term Memory
GP	Gaussian Process
DL	Deep Learning
MAE	Mean Average Error
GAN	Generative Adversarial Network
GRU	Gated Recurrent Unit
LSTM	Long-Short Term Memory
ML	Machine Learning
MAP	Maximum A Posteriori
MSE	Mean Squared Error
PDE	Partial Differential Equation
ODE	Ordinary Differential Equation
NN	Neural Network
FID	Frechet Inception Distance
FVD	Frechet Video Distance
MCMC	Markov Chain Monte Carlo
PKN	Prior Knowledge Network
FDM	Finite Difference Methods
SST	Sea Surface Temperature
BCCE	Brightness Constancy Constraint Equation

CHAPTER 1

INTRODUCTION

1.1 Context and Motivation

Machine Learning (ML), under the broader domain of Artificial Intelligence (AI), has recently gained the attention of the scientific community and of the private sector due to its newly acquired capability of solving complex problems. Advances in computer hardware and software allowed techniques that estimate a huge number of parameters to be exploited in a feasible time frame, contributing to the expansion of this area of study. At the annual ImageNet classification challenge held in 2012, traditional techniques have been outperformed by Deep Learning (DL) method. Given enough training data, deep networks can learn a meaningful representation of the image together with the classification. Since then, DL-based models have shown outstanding results in various machine learning tasks such as object detection, regression, image inpainting or forecasting. This opens the field for even more challenging applications involving complex phenomena, where deep networks can be designed to handle the modeling of physical processes. These complex problems are tackled by both academic and industrial actors, both in ML and physical science communities, which makes this research domain a rich and extremely active field.

With the availability of very large datasets captured via different types of sensors, the physical modeling paradigm is being challenged by the statistical ML paradigm, which offers a prior-agnostic approach. However, despite impressive successes in a variety of domains as demonstrated by the deployment of Deep Learning methods in fields such as vision, language, speech, etc., the statistical approach is not yet ready to challenge the physical paradigm for modeling complex natural phenomena, or at least it has not demonstrated how to. This is a new challenge for this field and an emerging research direction in the ML community. We believe that knowledge and techniques accumulated for modeling physical processes in well-developed fields such as maths or physics could be useful as a guideline to design efficient learning systems and conversely, that the ML paradigm could open new directions for modeling such complex phenomena.

This thesis is motivated by physical processes, and more precisely how we can incorporate physical prior knowledge to model them. We focus on specific aspects of physical processes : spatio-temporal forecasting, hidden state

discovery, and observation recovery. All this aspect can also be seen as the study of the dynamics of the process.

A physical process is a sustained phenomenon marked by gradual changes through a series of states occurring in the physical world. Physicists and environmental scientists attempt to model these processes in a principled way through analytic descriptions of the scientist's prior knowledge of the underlying processes. Conservation laws, physical principles or phenomenological behaviors can be formalized by using differential equations.

Dynamical systems are a tool of choice to model the evolution of phenomena occurring in nature. In order to derive a dynamical system describing a real world physical process, one must first gather measurements of this system. Then, a set of variables X_t describing the system at a given time t, called the *state*. The state is defined as the set of variables (called state variables) so that the knowledge of these variables at time t_0 is sufficient to describe the behavior of the system for any time $t \geq t_0$. Together with the knowledge of a function F, we can consider an evolution equation of the form :

$$\frac{dX_t}{dt} = F(X_t) \tag{1.1}$$

Many phenomena studied in physics, computer vision, biology, etc., obey a general equation of this form. For this reason, an extensive effort has been put into gaining a better understanding and resolving this equation. However, for many practical problems, the relationship between the components of the state is highly non-linear and complex to describe it analytically: finding an appropriate evolution model F can be a difficult problem.

For many real-world applications, the entire state of the system is often not fully visible to external sensors or corrupted. The state is said to be partially observable. *e.g.*, when studying the ocean's circulation, variables contained in the system's state such as surface temperature or salinity are observable through satellite, while others subsurface variables are not systematically observed. In other words, the available data are only a projection of the complete state X_t. This observation process can be modeled with an operator $\mathcal{H}_t$ linking the system's state X_t to the corresponding observation Y_t:

$$Y_t = \mathcal{H}_t(X_t). \tag{1.2}$$

As mentioned before, despite the undeniable Deep Learning success, a fully data-driven approach is not yet ready to challenge the classical approach for modeling dynamical systems. We will try to demonstrate in this thesis that knowledge and techniques accumulated for modeling dynamical systems processes in well-developed fields such as maths or physics could be useful as a guideline to design efficient learning systems and conversely, that the ML paradigm could open new directions for modeling such complex phenomena.

We describe in the following section, three tasks that are relevant to the study and modeling of Deep Learning and Dynamical System : Forecasting, hidden state discovery and unsupervised signal recovery.

1.2 Tasks

1.2.1 Forecasting

Forecasting is the task of making future prediction based on past data. For a dynamical system, it consists in finding the function F and the parameters θ such that:

$$\frac{dX_t}{dt} = F_\theta(X_t), \tag{1.3}$$

Under the constraint that X is only partially observable. In Chapter 3, we consider the use of Deep Learning methods for forecasting complex phenomena, like those occurring in natural physical processes. Using an example application, namely Sea Surface Temperature Prediction, we show how general background knowledge gained from the physics, under the form of a Partial Differential Equation (PDE) could be used as a guideline for designing efficient Deep Learning models. In this example : $X_t = (I_t, w_t)$ where I_t is a temperature image and w the motion vector. In this setting, the temperature I_t is an input of the problem, while w is the is unknown. The function $\mathcal{H}$ is in this case : $\mathcal{H}(X_t) = I_t$. We propose in this chapter a "hybrid" Deep Learning model in order to tackle this problem.

In order to motivate the approach and to assess its generality, we demonstrate the existence of a formal link between the solution of a class of differential equations underlying a large family of physical phenomena and the proposed model, namely the advection diffusion equation. Experiments and comparison with series of baselines including a state-of-the-art numerical approach is then provided.

To the best of our knowledge, this work in one of the first works attempting to incorporate a PDE directly into a Deep Learning forecasting models. We have shown the pertinence of such an approach by comparing our approach to other approaches, and paved the way to a hybrid approach of forecasting systems.

Emmanuel de Bezenac, Arthur Pajot, and Patrick Gallinari (Feb. 2018). “Deep Learning for Physical Processes: Incorporating Prior Scientific Knowledge”. In: URL: https://openreview.net/forum?id=By4HsfWAZ.

1.2.2 Hidden State Discovery

We consider the problem of learning to model the evolution of physical processes evolving in space and in time, given only partial observations of the state. Formally, in this case, the observation operator $\mathcal{H}$ is a projection operator.

In Chapter 4, we propose a framework, where the system's dynamic is modeled by an unknown dynamical system, and the evolution term is estimated from the data. Given an initial state, an ODE solver can then be used to compute any future complete state, allowing retrieving the unobserved part of the state as well as forecasting future observations. We qualitatively and quantitatively study the results of our method over simulations of complex data.

Instead of learning a Residual Network (He et al. 2015) or a recurrent model, we chose to model the forecasting network as a dynamical system. This has two purposes : ensure the physical plausibility of the model by allowing it to explicitly handling hidden, but physical, variables (see Section 1.2.2), and thus improving its performance.

Discovering a hidden state is a crucial problem for physical sciences. When observing a complex phenomenon, there are often unknown underlying variables of interest. We show that our method learns to closely reproduce the unobserved dynamics of the state without direct supervision on the latter when the true initial state is given as input, and consistently outperform the baselines.

We also show that our method can still be successfully applied when the initial state is not available and that it produces an interpretable hidden state even in this case. This led to an arxiv paper :

Ibrahim Ayed, Emmanuel de Bézenac, Arthur Pajot, Julien Brajard, and Patrick Gallinari (Feb. 2019). "Learning Dynamical Systems from Partial Observations". In: *arXiv:1902.11136 [physics]*. URL: http://arxiv.org/abs/1902.11136

1.2.3 Inverse Problem

Many real world applications require acquiring information about the state of some physical system from incomplete and inaccurate measurements. For example, in infrared satellite imagery, one has to deal with the presence of clouds and a variety of other external factors perturbing the acquisition of temperature maps. This raises questions on how to recover the correct information and eliminate the contribution of external factors hindering the overall signal acquisition.

We introduce an observation function F such that we can simulate its measurement by first sampling θ from p_Θ and underlying distribution representing

the factor of corruption, and then computing $f(x;\theta)$. In a dynamical system setting, we can write :

$$Y_t = \mathcal{H}_t(X_t) = f(X_t, \theta),$$

with Y_t the observation and X_t the state of the system. In Chapter 5, we address the problem of recovering an underlying signal from lossy, inaccurate observations in an unsupervised setting. Typically, we consider situations where there is little to no background knowledge on the structure of the underlying signal, no access to signal-measurement pairs, nor even unpaired signal-measurement data. This is the case in physical system as we only have access to observation of the system. The only available information is provided by the observations and the measurement process statistics. By using Generative Adversarial Network (GAN), a Deep Learning model designed to produce signals that are indistinguishable from an input signal distribution, we cast the problem as finding the *maximum a posteriori* estimate of the signal given each measurement, and propose a general framework for the reconstruction problem. We use a formulation of GAN, where the generator takes as input a corrupted observation in order to produce realistic reconstructions, and add a penalty term tying the reconstruction to the associated observation. We evaluate our reconstructions on several image datasets with different types of corruptions. The proposed approach yields better results than alternative baselines, and comparable performance with model variants trained with additional supervision. This is one of the first works proposing to learn an inverse problem from a dataset of corrupted signals, without supervision.

We also present, for the special case of inpainting, two natural extensions.

In the first extension, we consider stochastic inpainting. Image completion or inpainting is generally a complex inverse problem, it is usually under specified so that there is not a unique solution. Most approaches propose a unique image reconstruction among all the possible ones. In this chapter, we solve a more challenging task that consists in learning the distribution of the plausible reconstructions.

In the second extension, we consider the case of incomplete sequence of observations. We propose a temporal extension of our model, by augmenting our Neural Network in order to handle video.

This led to two publications :

Arthur Pajot, Emmanuel de Bezenac, and Patrick Gallinari (Sept. 2018). "Unsupervised Adversarial Image Reconstruction". In: URL: https://openreview.net/forum?id=BJg4Z3RqF7 Yuan Yin, Arthur Pajot, Patrick Gallinari, and Em-

manuel de Bézenac (Aug. 2019). "Unsupervised Inpainting for Occluded Sea Surface Temperature Sequences". In: *Climatinformatics Workshop*

1.3 Outline of the thesis

We begin this manuscript by introducing some notions important for our work in Chapter 2. Without making an exhaustive review, we will explain key concepts on forecasting, differential equations and adversarial networks. We'll then address the three different tasks : the forecasting in Chapter 3, the hidden state discovery in Chapter 4 and the state reconstruction in Chapter 5. We'll then conclude this thesis in Chapter 6 where we'll introduce some possible future work.

CHAPTER 2

RELATED WORKS

Contents

2.1 Forecasting 7
2.1.1 Time Series Forecasting 8
2.1.2 Recurrent Neural Networks 9
2.1.3 Video Prediction 10
2.2 Modeling Differential Equations 13
2.2.1 Ordinary Differential Equations 13
2.2.2 Partial Differential Equations 16
2.2.3 PDE and Neural Networks 18
2.3 Inverse Problem 18
2.3.1 GAN 18
2.3.2 Image Translation 21
2.3.3 Inverse Problem 23

In this chapter, we will introduce some notions critical for our work. Without making an exhaustive review, we will explain key concepts that will allow us to clarify and highlight the contributions made in this thesis.

First in Section 2.1 we provide a big picture on non-physical spatiotemporal forecasting. Then in Section 2.2 we will introduce some key concepts about Partial Differential Equation (PDE) and Neural Networks. Finally, in Section 2.3 we will discuss GAN, image translation and unsupervised Inverse Problem.

2.1 Forecasting

In this section, we briefly introduce the problem of classical time series forecasting. The Deep Learning community recently attacked the problem of video prediction. Since this bears some similarities with our problem, we will briefly explain some core concepts of this field.

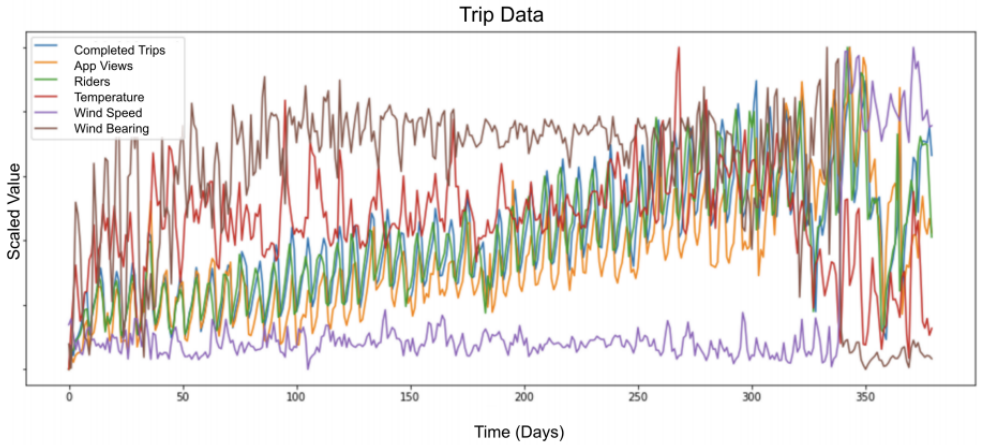


Figure 2.1 – **Examples of a time serie.** Taken from "Time-series Extreme Event Forecasting with Neural Networks at Uber".

2.1.1 Time Series Forecasting

2.1.1.1 Classical Time Series

Data collected on the internet, or on the physical world, can often be seen as a time series: the measurements of a process are often realized at different times, allowing the monitoring of the process's evolution.

For instance, in the medical field, the information measured by the electroencephalogram (EEG) or by electrocardiogram, or data representing gene expression (Aach et al. 2001), or even the growth data of an individual represent all a time series. Temporal time series are found, in several fields such as finance, weather forecast, signal and speech processing. Time series models can be viewed as dynamical models for temporal data.Let us first define some basic functions of a discrete-time sequence of real-valued random variables $\{Y_t : t \in D_t\}$. We assume $D_t = \{0, 1, \ldots\}$, and we refer to $\{Y_t : t = 0, 1, \ldots\}$ as a time series.

Forecasting a time series consists in, given an input $\{Y_t : t \in [0 : T]\}$, predicting the following k values $\{Y_t : t \in [T+1 : T+k]\}$.

Time series forecasting is often performed by a regression function, which is a model that receives a sequence of observations and returns a scalar or a vector of real numbers. The regression function that predicts the value at the next instant of time $t+1$, given a sequence of values in $Y_{0:t}$ is expressed by Equation 2.1.

$$Y_{t+1} = F_\theta(Y_{t-k:t}) \tag{2.1}$$

where k is a parameter of the regression. Modeling a time series consists in finding the parameters θ parametrizing the function F.

The classical topic of time series modeling and forecasting has given rise to an extensive literature. In statistics, classical linear models include many variations around autoregressive and moving average models. In machine learning, non-linear extensions of these models based on neural networks have been proposed as early as the 90s, opening the way to many other non-linear models including kernel methods (Thissen et al. 2003).

2.1.1.2 Spatio-Temporal Time Series

Relational time series have mainly been studied in the field of spatiotemporal statistics (Cressie et al. 2015; Wikle et al. 2010). The traditional method first relied on a descriptive approach using the first- and second-order moments of the process for modeling the spatiotemporal dependencies. More recently, dynamic state models, where the current state is conditioned on the past, have been explored (Cressie et al. 2015). These models have been considered both for continuous/discrete space and time components. However, the most common way is to consider a discrete time, leading to the modeling of time series of spatial processes as we study here. When space is discrete, the model comes down to a general vectorial autoregressive formulation. These models face a curse of dimensionality in the case of a large number of sources. Different strategies have been adopted to solve this problem such as embedding the spatiotemporal process in a low-dimensional manifold or parameter reduction (Cressie et al. 2015), leading to model families quite similar to the ones used in machine learning for modeling dynamic phenomena.

2.1.2 Recurrent Neural Networks

Recently, there has been a growing interest in learning latent representation through neural networks and deep learning. Dynamical state space models such as Recurrent Neural Network (RNN), which have been used for time series forecasting in different contexts since the early nineties (Connor et al. 1994), have witnessed important successes in different areas for general sequence modeling problems, leading to breakthroughs in domains like speech (Graves et al. 2013), language generation (Sutskever et al. n.d.), translation (Cho et al. 2014), and many others.

RNN is a class of artificial neural network where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior. Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs.

$$h_t = F(W * x_t + U * h_{t-1}) \tag{2.2}$$

One can build a deep recurrent neural network by simply stacking units. A simple recurrent neural network works well only for a short-term memory. It suffers from a fundamental problem if we have a longer time dependency. As we go back to the lower layers, gradient often gets smaller, eventually causing weights to never change at lower layers. This is a problem because we want our RNNs to forecast long-time series, which involves keeping track of long sequences of observations.

Long-Short Term Memory (LSTM) are a special kind of RNN, capable of learning long-term dependencies. They were introduced in (Hochreiter et al. 1997) , and were refined and popularized by many people (Dey et al. 2017). They work tremendously well on a large variety of problems, and are now widely used. LSTMs are explicitly designed to avoid the long-term dependency problem. Their equations are :

$$\begin{aligned} i_t &= \sigma(W_{xi} * x_t + W_{hi} * h_{t-1} + W_{ci} \circ C_{t-1} + b_i) \\ f_t &= \sigma(W_{xf} * x_t + W_{hf} * h_{t-1} + W_{cf} \circ C_{t-1} + b_f) \\ C_t &= f_t \circ C_{t-1} + i_t \circ \tanh(W_{xc} * x_t + W_{hc} * h_{t-1} + b_c), \\ o_t &= \sigma(W_{xo} * x_t + W_{ho} * h_{t-1} + W_{co} \circ C_t + b_o) \\ h_t &= o_t \circ \tanh(C_t) \end{aligned} \tag{2.3}$$

Here, i, f, o are the input, forget and output gates, respectively. Note that they share the exact same equations, with different parameter matrices. W_h is the recurrent connection at the previous hidden layer and current hidden layer, W_x is the weight matrix connecting the inputs to the current hidden layer). The input gate defines how much of the newly computed state for the current input you want to let through. The forget gate defines how much of the previous state you want to let through. Finally, The output gate defines how much of the internal state you want to expose to the external network (higher layers and the next time step).h is the internal memory of the unit. It is a combination of the previous memory, multiplied by the forget gate, and the newly computed hidden state, multiplied by the input gate. Thus, intuitively it is a combination of how we want to combine previous memory and the new input. h_t is the hidden state, computed by multiplying the memory with the output gate. Not all of the internal memory may be relevant to the hidden state used by other units in the network. Intuitively, plain RNNs could be considered a special case of LSTMs. An LSTM unit is described in Figure 2.2.

2.1.3 Video Prediction

It is only recently that video prediction emerged as a task in the Deep Learning community. For this task, people are generally interested in predicting accurately the displacement/ emergence/ disappearing of objects in videos.

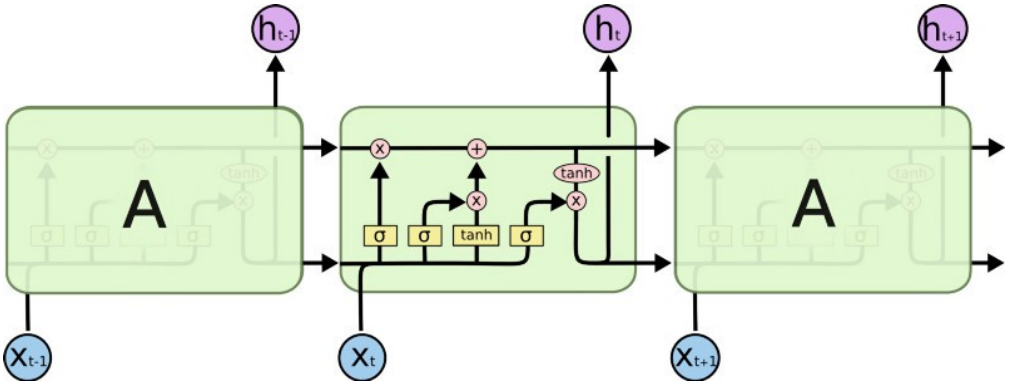


Figure 2.2 – **A schematic LSTM system**. *A* is some LSTM cell. Image taken from : http://colah.github.io/posts/2015-08-Understanding-LSTMs/

In our application, the goal is clearly different since we are interested into modeling the whole dynamics behind image changes and not into following a moving object.

Let us first introduce some methods that perform prediction by computing optical flow or a similar transformation. Classical methods rely on the Brightness Constancy Constraint Equation (BCCE) (Equation 2.4), derived from the observation that surfaces usually persist over time and hence the intensity value of a small region remains the same despite its position change (Sun et al. 2008).

Our work takes inspiration from recent optical flow methods, applying them to forecasting future images, as opposed to usual applications where the goal is motion estimation. The general approach to retrieving motion is to make assumptions on how elements in the video get displaced in time, transcribe these hypotheses into equations, and minimize energy functional enforcing these equations.

$$I(x,t) = I(x+\Delta x, t+\Delta t) \tag{2.4}$$

Optical Flow Estimation (Dosovitskiy et al. 2015) formulate optical flow as a supervised regression problem, using a Convolutional Neural Network (CNN) to predict motion. (Ilg et al. 2016) build on this approach and propose to use an ensemble of these CNN architectures. They assess results on par with state-of-the-art methods for optical flow, while maintaining a small computational overhead. The difficulty here is that these methods require a notable quantity of target data, i.e., optical flow images, while because of the complexity of manually annotating flow images, there are only a few small annotated collections available. (Dosovitskiy et al. 2015; Ilg et al. 2016) chose to pretrain their model on a synthetic dataset made of computer animations and their associated motion

and they show that this information transfers well to real videos. (J. J. Yu et al. 2016) demonstrate that it is possible to predict the optical flow between two input images in an unsupervised way using a CNN and a Spatial Transformer Network (Jaderberg et al. 2015).

Video Prediction With Optical Flow Now we briefly describe related work that tackles the task of video prediction with the help of optical flow. Both (Patraucean et al. 2015) and (Finn et al. 2016) use some form of motion flow estimation. For next frame prediction (Patraucean et al. 2015) introduce a Spatial Transformer Network (STN) module at the hidden layer of a LSTM in order to estimate a motion field in this latent space. The resulting image is then decoded in the original image space for prediction. This approach clearly does not allow introducing prior knowledge on the field vector as this has been done in our work. (Finn et al. 2016) learn affine transformations on image parts in order to predict the object displacement and (Van Amersfoort et al. 2017) proposed a similar model.

Direct Next Frame Prediction Let us now consider models that directly attempt to predict the next frame without estimating a motion field.

In (Mathieu et al. 2015) the authors, train a convolutional network to generate future frames given an input sequence. To deal with the inherently blurry predictions obtained from the standard Mean Squared Error (MSE) loss function they propose an adversarial training method. It is the first Deep Learning attempt to directly predict the next frame of a video.

Significant results have been obtained with the Video Pixel Network of (Kalchbrenner et al. 2016) which is a sophisticated architecture composed of resolution preserving CNN encoders, LSTM and PixelCNN decoders which form a conditional Spatio-temporal video autoencoder with differentiable memory. They reach a high accuracy on moving MNIST and good performance on a robot video dataset. A drawback is the complexity of the model and the number of parameters: they must use respectively 20M and 1M training frames on these two datasets.

The major drawback of LSTM in handling spatiotemporal data is its usage of full connections in input-to-state and state-to-state transitions in which no spatial information is encoded. Although the LSTM layer has proven powerful for handling temporal correlation, it contains too much redundancy for spatial data. To address this problem, the authors in (X. Shi et al. 2015) propose ConvLSTM which has convolutional structures in both the input-to-state and state-to-state transitions. An improvement of ConvLSTM is PredRNN++ (Y. Wang et al. 2018), which is a very competitive approach in Video Prediction.

2.2 Modeling Differential Equations

In this section we describe how Machine Learning, and more precisely Neural Network have been used to model Differential Equations. We first introduce briefly what Ordinary Differential Equation (ODE) are, as well as their link dynamical systems and neural networks. We then describe PDE, their discretization and Deep Learning attempts to model them.

In their reference book (Cressie et al. 2015) advocate the use of physical background knowledge to build statistical models. They show how the design of statistical models can be inspired from partial differential equations linked to an observed physical phenomenon. They mainly consider autoregressive models within a hierarchical Bayesian framework. However those approaches often focus on local, small scale, phenomena and use Markov Chain Monte Carlo (MCMC) techniques, so that they cannot take into account large quantities of data. Another interesting research direction is the use of Neural Network (NN)s for reducing the complexity of numerical simulations for physical processes. Generally, in these approaches statistical models are used in place of a computational demanding component of the numerical simulation process.

2.2.1 Ordinary Differential Equations

In mathematics, an ordinary differential equation (ODE) is a differential equation containing one or more functions of one independent variable and the derivatives of those functions.

2.2.1.1 Dynamical System

Let us now focus on the particular case of a dynamical system.

$$\forall t,\ \frac{dX_t}{dt} = F(X_t)$$

This problem can be approached by considering Neural Network $\{F_\theta\}$, and looking for parameters θ such that it approach the solution X^θ of:

$$\frac{dX_t}{dt} = F_\theta(X_t) \tag{2.5}$$

Typically, we can think about neural networks as a series of discrete layers, each one taking as input a previous state vector $\mathbf{h}_n$ and to produce a new state vector $\mathbf{h}_{n+1} = F(\mathbf{h}_n)$. Here, let us assume that each layer is the same width (i.e., $\mathbf{h}_n$ and $\mathbf{h}_{n+1}$ have the same dimension, for every n). Each hidden state depends on a neural layer, S, which itself depends on parameters, where t is the layer depth.

$$h_{t+1} = F_\theta(h_t)$$

Residual Neural Networks (ResNet) have achieved state-of-the-art results in several vision tasks (He et al. 2015; Ledig et al. 2016). The core concept of ResNet is the idea of "Shortcut Connection" that skip layers by adding the output of the last layer to the next layer :

$$h_{t+1} = h_t + F_\theta(h_t) = h_t + \frac{\Delta t}{\Delta t} F_\theta(h_t) = h_t + \Delta t G(h_t),$$

which can be seen as a Euler discretization of an ODE (see Section 2.2.2.2) with $G = \frac{F_\theta}{\Delta t}$.

2.2.1.2 Neural ODE

Chen et al. 2018 consider the continuous limit of each discrete layer in the network. Instead of a discrete number of layers between the input and output domains, this allows the progression of the hidden states to become the following differential equation (ODE):

$$\forall t, \ \frac{dh(t)}{dt} = F(t, h(t), \theta_t), \tag{2.6}$$

where $h(t)$ is the value of the hidden state evaluated for some t, which we understand as a continuous parametrisation of layer depth. Given an initial condition $h(0)$, the output from an neural network, can be specified as the solution to Equation 2.6 as time T.

$$h(T) = h(0) + \int_t^T \frac{dh(t)}{dt} dt = h(0) + \int_t^T F(t, h(t), \theta_t) dt$$

.

This value can be computed by a black-box differential equation solver, which evaluates the hidden unit dynamics F or by. An overview of Neural ODE is shown in Figure 2.3.

An ODE Network (our black box solver) has two main applications : the first is Normalizing Flow. It is a promising class of generative models that maps points from a simple distribution to a complex distribution through an invertible neural network. Likelihood-based training of these models requires restricting their architectures to allow cheap computation of Jacobian determinants. In (Grathwohl et al. 2018) the authors propose a continuous-time invertible ODE generative model which allows unrestricted neural network architectures. The second one is the learning of Dynamical Systems that we will develop in more details in Chapter 4.

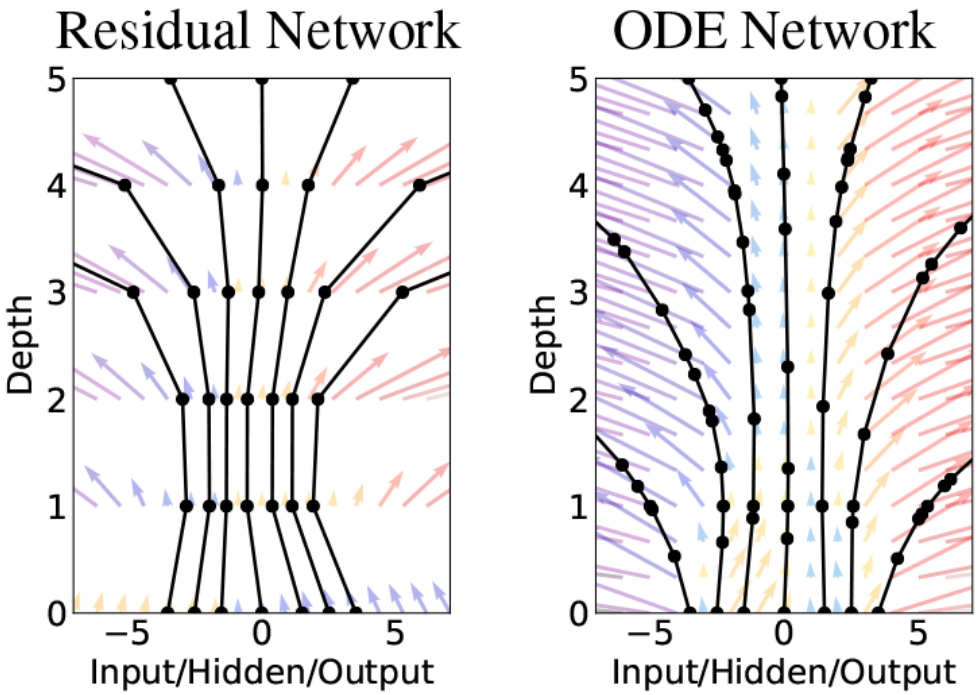


Figure 2.3 – *Left* A Residual Network defined with a fixed number of layers where the information transforms sequentially with each layer. *Right* An ODE Net with continuous layers, the information transforms continuously through the layers. The black circles represent evaluation location. Figure taken from Chen et al. 2018

2.2.1.3 Adjoint Method

The main difficulty for using a neural network parametrization of an ODE is backpropagating through the ODE solver. Despite numerical solvers being often differentiable, making the backpropagation step straightforward, the memory cost is high and additional numerical errors often occurs. In order to perform backpropagation the gradient of the loss function with respect to all parameters must be computed. As different numerical solver may take a varying number of steps when numerically integrating between two-time points, a general method for computing the gradient of the loss at each intermediate step is required.

Chen et al. 2018 introduce a solution to this problem using the adjoint sensitivity method which can be used regardless of the choice of the differentiable ODE solver and with constant memory consumption. This section gives an overview of how the adjoint sensitivity method for backpropagation works. A proof of this method is given by Chen et al. in appendix B of their paper (Chen et al. 2018).

The adjoint method works by constructing the adjoint state $a(t) = \frac{dL}{dh(t)}$, where L is the loss function $h(t)$ is the output after each step taken by the Numerical Solver which follows the differential equation equation 2.6. It can be shown that the adjoint state follows the differential equation :

$$\frac{da(t)}{dt} = -a(t)\frac{\partial F(h(t), t, \theta)}{\partial h(t)}. \tag{2.7}$$

We can then solve the differential equation equation 2.7 backwards in time, i.e., from the final output time t_N to the starting time t_0, similar to regular backpropagation. Hence we acquire the gradients with respect to the hidden state at any time as:

$$\frac{dL}{dh(t_0)} = a(t_0) = a(t_N) + \int_{t_N}^{t_0} \frac{da(t)}{dt} dt.$$

and

$$\frac{dL}{dh(t_n)} = a(t_N) + \int_{t_N}^{t_0} \frac{da(t)}{dt} dt = a(t_N) - \int_{t_N}^{t_0} \frac{\partial F(h(t), t, \theta)}{\partial h(t)} dt. \tag{2.8}$$

When using multiple steps in an ODE solver, we can simply integrate backwards in time between each time step taken in order to get the gradient at all the different time steps, i.e., from t_N to t_{N-1}, then from t_{N-1} to t_{N-2} and so on, and summing the gradients after each solved step.

2.2.2 Partial Differential Equations

2.2.2.1 PDE

PDE are equations that involve rates of change with respect to continuous variables. A general form of a PDE is :

$$F\left(x_1, \dots x_n; u, \frac{\partial u}{\partial x_1}, \dots \frac{\partial u}{\partial x_n}; \frac{\partial^2 u}{\partial x_1 \partial x_1}, \dots \frac{\partial^2 u}{\partial x_1 \partial x_n}; \dots\right) = 0. \tag{2.9}$$

Where $u = u(x_1, x_2, \dots, x_n)$ is the unknown function and $F(\dots)$ is a given function.

A simple example is the 1-d heat equation

$$\frac{\partial u}{\partial t} - k\frac{\partial^2 u}{\partial x^2} = 0. \tag{2.10}$$

Where $u = u(x, t)$.

2.2.2.2 Numerical Scheme

Discretization is the process of transferring continuous functions, or PDE into discrete counterparts.

Finite Difference Methods (FDM) are numerical methods for solving differential equations by approximating them with difference equations, in which finite differences approximate the derivatives. It consists in approximating the differential operator by replacing the derivatives in the equation using differential quotients.

For the sake of simplicity, we shall consider the one-dimensional case only. The main concept behind any finite difference scheme is related to the definition of the derivative of a smooth function u at a point $x \in \mathbb{R}$. The simplest approximation of $u_t = f(u,t)$ is to discretize the time derivative u_t by $\frac{u_{n+1}-u_n}{\Delta t}$ and approximate the right and side by $f(u_n, t_n)$. This leads to the forward *explicit Euler scheme*.

$$u_{n+1} = u_n + \Delta t f(u_n, t_n) \tag{2.11}$$

.

Where Δt is a fixed step size.

Runge Kuta This method lies on the iteration principle : the first estimation is used to compute the second, which is more accurate, and so on. The Order-1 Runge Kutta is similar to the forward Euler scheme. The second order is a composition of the Euler method :

$$u_{n+1} = u_n + \Delta t f\left(t_n + \frac{\Delta t}{2}, u_n + \frac{\Delta t}{2} f\left(u_n, t_n\right)\right).$$

This method estimates the derivative in the middle of the integration step :

$$\begin{aligned} u_{n+\frac{1}{2}} &= u_n + \frac{\Delta t}{2} f\left(u_n, t_n\right) \\ u'_{n+\frac{1}{2}} &= f\left(u_{n+\frac{1}{2}}, t_n + \frac{\Delta t}{2}, \right) \end{aligned}$$

and then integrating one more step from this estimation :

$$u_{n+1} = u_n + \Delta t u'_{n+\frac{1}{2}}.$$

We can use Runge Kutta of arbitrary order, by considering higher order of integration steps.

2.2.3 PDE and Neural Networks

2.2.3.1 Data-Driven Simulation of fluid Dynamic.

An interesting research direction is the use of NNs for reducing the complexity of numerical simulations for physical processes. Generally, in these approaches, statistical models are used in place of a computational demanding component of the numerical simulation process.

For example, in the domain of fluid dynamics, (Tompson et al. 2017) and (Ladický et al. 2015) propose to use regressors for simulating fluid and smoke animation. (Ladický et al. 2015) use a random forest to compute particle location and (Tompson et al. 2017) use a CNN to approximate part of a numerical PDE scheme. In these approaches, Machine Learning (ML) is only a component of a numerical simulation scheme whereas we will aim at modeling the whole physical process via a Deep Learning approach.

2.2.3.2 Data-Driven Discovery of Differential Equations.

In the past, several works have already attempted to learn differential equations from data, such as e.g., (Crutchfield et al. 1987; Alvarez et al. 2013). More recently, (Rudy et al. 2017) uses sparse regression on a dictionary of different terms to recover the underlying PDE. In (Raissi et al. 2017), they propose recovering the coefficients of the differential terms by deriving a Gaussian Process (GP) kernel from a linearized form of the PDE. (Long et al. 2018) carefully tailor the neural network architecture, based on the discretization of the different terms of the underlying PDE. (Raissi 2018) develops a NN framework for learning PDEs from data. (Fablet et al. 2017) constructs a bilinear network and uses an architecture similar to finite difference schemes to learn fully observed dynamic systems. In those approaches, we often see that the form of the variable dependency is supposed to be known and that the context is the unrealistic setting where the state is fully observed. Other approaches were proposed in case of partially observed fields relying on a combination of data assimilation and machine learning (Bocquet 2012).

2.3 Inverse Problem

2.3.1 GAN

In this section we describe Generative Adversarial Network (GAN), an extremely popular generative modeling tool, with applications to computer vision. A GAN is a generative model allowing producing samples from a distribution, given a large number of examples.

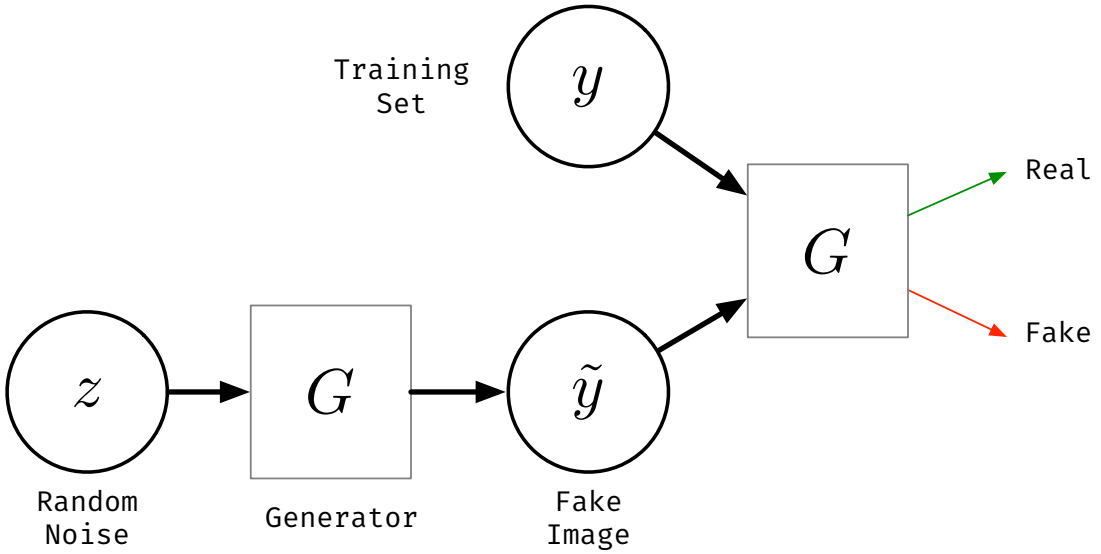


Figure 2.4 – The GAN framework pits two adversaries against each other in a game. Each player is represented by a differentiable function controlled by a set of parameters. Typically these functions are implemented as deep neural networks.

The basic idea of GANs is to set up a game between two networks. One of them is called the generator. The generator creates samples that are intended to come from the same distribution as the training data. The other network is the discriminator. The discriminator examines samples to determine whether they are real or fake. The discriminator learns using traditional supervised learning techniques, dividing inputs into two classes (real or fake). The generator is trained to fool the discriminator.

The game plays out in two scenarios. In one scenario, training examples $\boldsymbol{x}$ are randomly sampled from the training set and used as input for the first player, the discriminator, represented by the function D. The goal of the discriminator is to output the probability that its input is real rather than fake. In this first scenario, the goal of the discriminator is for $D(\boldsymbol{x})$ to be near 1.

In the second scenario, inputs $\boldsymbol{z}$ to the generator are randomly sampled from the model's prior over the latent variables. The discriminator then receives input $G(\boldsymbol{z})$, a fake sample created by the generator. In this scenario, both players participate. The discriminator strives to make $D(G(z))$ approach 0 while the generative strive to make the same quantity approach 1. If both models have sufficient capacity, then the Nash equilibrium of this game corresponds to the $G(\boldsymbol{z})$ being drawn from the same distribution as the training data, and $D(\boldsymbol{x}) = \frac{1}{2}$ for all $\boldsymbol{x}$.

The two players in the game are represented by two functions, each of which is differentiable both with respect to its inputs and with respect to its parameters. The discriminator is a function D that takes $\boldsymbol{x}$ as input and uses

$\boldsymbol{\theta}^{(D)}$ as parameters. The generator is defined by a function G that takes $\boldsymbol{z}$ as input and uses $\boldsymbol{\theta}^{(G)}$ as parameters. Both players have cost functions that are defined in terms of the two players' parameters. In the original formulation (Goodfellow et al. 2014) the discriminator wishes to minimize the discriminator loss : $\mathcal{L}^{(D)}\left(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}\right)$ and must do so while controlling only $\boldsymbol{\theta}^{(D)}$. The generator wishes to minimize the generator loss : $\mathcal{L}^{(G)}\left(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}\right)$ and must do so while controlling only $\boldsymbol{\theta}^{(G)}$. Because each player's loss depends on the other player's parameters, but each player cannot control the other player's parameters, this scenario is most straightforward to describe as a game rather than as an optimization problem. The solution to an optimization problem is a (local) minimum, a point in parameter space where all neighboring points have greater or equal cost. In this context, a Nash equilibrium is a tuple $(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)})$ that is a local minimum of $\mathcal{L}^{(D)}$ with respect to $\boldsymbol{\theta}^{(D)}$ and a local minimum of $\mathcal{L}^{(G)}$ with respect to $\boldsymbol{\theta}^{(G)}$.

The Training Process The training process consists in a simultaneous gradient descent. On each step, two minibatches are sampled: a minibatch of $\boldsymbol{x}$ values from the dataset and a minibatch of $\boldsymbol{z}$ values drawn from the model's prior over the latent variables. Then two gradient steps are performed simultaneously: one updating $\boldsymbol{\theta}^{(D)}$ to reduce $\mathcal{L}^{(D)}$ and one updating $\boldsymbol{\theta}^{(G)}$ to reduce $\mathcal{L}^{(G)}$. In both cases, it is possible to use any gradient-based optimization algorithm .

2.3.1.1 Cost functions

Several different cost functions may be used within the GAN's framework.

Standard GAN Loss In (Goodfellow et al. 2014), the authors introduce the following loss :

$$\begin{aligned}\mathcal{L}^{(D)}(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}) &= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\big[log(D(x))\big] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\big[log(1 - D(G(z)))\big] \\ \mathcal{L}^{(G)}(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}) &= \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\big[log(D(G(z)))\big].\end{aligned} \tag{2.12}$$

Here, the generator loss $\mathcal{L}^{(G)}$ is minimized and the discriminator loss $\mathcal{L}^{(D)}$ is maximized.

Least Squares Generative Adversarial Networks When updating the generator, this loss function will cause the problem of vanishing gradients for the samples that are on the correct side of the decision boundary, but are still far from the real data. To remedy this problem, the Least Squares Generative Adversarial Networks (LSGANs) was proposed (Mao et al. 2017) :

$$\begin{aligned}\mathcal{L}^{(D)}(\boldsymbol{\theta}^{(D)},\boldsymbol{\theta}^{(G)}) =& \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[(D(\boldsymbol{x})-1)^2\right] + \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[D(G(\boldsymbol{z}))^2\right] \\ \mathcal{L}^{(G)}(\boldsymbol{\theta}^{(D)},\boldsymbol{\theta}^{(G)}) =& \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-1)^2\right].\end{aligned} \quad (2.13)$$

Here, the generator loss $\mathcal{L}^{(G)}$ and the discriminator loss $\mathcal{L}^{(D)}$ are minimized.

Hinge GAN An alternative loss, which has gained large success (Zhang et al. 2018; "Large Scale GAN Training for High Fidelity Natural Image Synthesis" n.d.) is the Hinge Loss :

$$\begin{aligned}\mathcal{L}^{(D)}(\boldsymbol{\theta}^{(D)},\boldsymbol{\theta}^{(G)}) =& -\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[max(0,-1-D(\boldsymbol{x}))\right]+ \\ & \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[max(0,-1-D(G(\boldsymbol{z})))\right] \\ \mathcal{L}^{(G)}(\boldsymbol{\theta}^{(D)},\boldsymbol{\theta}^{(G)}) =& -\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[D(G(\boldsymbol{z}))\right].\end{aligned} \quad (2.14)$$

Here, the generator loss $\mathcal{L}^{(G)}$ and the discriminator loss $\mathcal{L}^{(D)}$ are minimized. There exists several other loss functions for GAN which won't be discussed here.

2.3.2 Image Translation

Recent works have achieved high quality results in image-to-image translation (Isola et al. 2016; Zhu et al. 2017a; Almahairi et al. 2018). Image translation consists in transforming images from a domain X to a domain Y. For example, inpainting where one wants to transform an incomplete image Y to complete image X is a special case of image translation.

Pix2Pix We could formulate the problem in a supervised setting where we have access to images to paired images from both domains. Pix2Pix (Isola et al. 2016) learns in a supervised manner by combining an adversarial loss with a L1 loss, thus requiring paired data samples. It uses adversarial networks to help produce perceptually realistic results. It trains a generator $G : Y \to X$ and discriminator D by formulating their objective as an adversarial game. The discriminator attempts to differentiate between real images from the dataset and fake samples produced by the generator. X and Y designate the above-mentioned image domain.

$$\mathcal{L}_{\text{GAN}}(G,D) = \mathbb{E}_{Y\sim p(Y)}[\log(D(Y))] + \mathbb{E}_{X\sim p(X)}[\log(1-D(G(X)))] \quad (2.15)$$

To encourage the output of the generator to match the input, they use a ℓ_1 loss between the output and the ground truth image.

$$\mathcal{L}_1^{\text{image}}(G) = \mathbb{E}_{X,Y\sim p(X,Y)}||X-G(Y)||_1 \quad (2.16)$$

The final loss function uses the GAN and ℓ_1 term, weighted by λ.

$$G^* = \arg\min_G \max_D \quad \mathcal{L}_{\text{GAN}}(G, D) + \lambda \mathcal{L}_1^{\text{image}}(G) \tag{2.17}$$

CycleGAN To alleviate the problem of obtaining data pairs, unpaired image-to-image translation frameworks (Zhu et al. 2017a; Almahairi et al. 2018) have been proposed. The principal contribution here is CycleGAN (Zhu et al. 2017a), which allows image translation by computing a bijection between two domains.

For the mapping function $G : X \to Y$ and its discriminator D_Y, the objective can be expressed as:

$$\mathcal{L}_{\text{GAN}}(G, D_Y) = \mathbb{E}_{Y \sim p(:)}[\log D_Y(Y)] + \mathbb{E}_{X \sim p(X)}[\log(1 - D_Y(G(X))] \tag{2.18}$$

where G tries to generate images $G(x)$ that look similar to images from the domain Y, while D_Y aims to distinguish between translated samples $G(x)$ and real samples y. We can write the same objective for the mapping function $F : Y \to X$ and its discriminator. D_X

Adversarial losses alone cannot guarantee that the learned function can map an individual input x_i to a desired output y_i. To further reduce the space of possible mapping functions the authors propose a *cycle consistency loss*.

$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p(X)}[\|F(G(x)) - x\|_1] \tag{2.19}$$
$$+ \mathbb{E}_{y \sim p(Y)}[\|G(F(y)) - y\|_1]. \tag{2.20}$$

The full objective is then:

$$\begin{aligned} \mathcal{L}(G, F, D_X, D_Y) =& \mathcal{L}_{\text{GAN}}(G, D_Y) \\ &+ \mathcal{L}_{\text{GAN}}(F, D_X) && (2.21) \\ &+ \lambda \mathcal{L}_{\text{cyc}}(G, F), && (2.22) \end{aligned}$$

where λ controls the relative importance of the two objectives. Cycle GAN aims to solve:

$$G^*, F^* = \arg\min_{G,F} \max_{D_x, D_Y} \mathcal{L}(G, F). \tag{2.23}$$

Augmented Cycle GAN (Almahairi et al. 2018) and Bicycle GAN (Zhu et al. 2017b) add stochasticity in the translation process by learning a mapping between a latent vector and the target domain. All these approaches then rely on some form of supervision either requiring paired or unpaired datasets.

2.3.3 Inverse Problem

At the heart of many signal processing tasks is a linear inverse problem, where the goal is to reconstruct a signal $x \in \mathbb{R}^d$ from a set of measurements $y \in \mathbb{R}^m$ of the form $y = Ax + n$, where $A \in \mathbb{R}^{m \times d}$ is the measurement operator and $n \in \mathbb{R}^m$ is the noise. For image inpainting, $\boldsymbol{y}$ is an image with masked regions and A is the linear operation applying a pixelwise mask to the original image $\boldsymbol{x}$; for super-resolution, $\boldsymbol{y}$ is a low-resolution image and the operation A downsamples high resolution images; in compressive sensing, $\boldsymbol{y}$ denotes compressive measurements and A is the measurement matrix. Similarly, non-linear inverse problems can be defined by the non-linear measurement function F, mapping measurement to complete signal x by $y = F(x)$.

A method to solve this problem is to learn a mapping from the domain of observed signals Y to the domain of signals X. This mapping is the *inverse* F^{-1} of the mapping F.

2.3.3.1 Inverse Problem and NN.

In the context of image super resolution (W. Shi et al. 2016, Sønderby et al. 2016, Mardani et al. 2017) attempt to retrieve *maximum a posteriori* estimates of the super-resolution image conditioned on an input image. They use a generative model of the signal trained in an adversarial fashion using samples from signal distribution to constrain their reconstructions. Their approach is fully supervised .

Other works attempt to solve ill-posed inverse problems using generative models (Bora et al. 2017; Asim et al. 2018; Tripathi et al. 2018; Van Veen et al. 2018). The general approach in all these papers consists to first train a GAN model on the uncorrupted image distribution. Then, given a measurement from which we wish to reconstruct the signal, the mapping is *inverted* by finding the latent input code that generated the uncorrupted image. This requires solving an optimization problem for each image, which takes several minutes (Van Veen et al. 2018) on GPU, and requires random restarts to avoid falling a bad local minimum. Again, the setting is fully supervised with uncorrupted images.

Finally, (Lehtinen et al. 2018) propose a method for denoising images without direct supervision. They train a network to regress a noisy image to the same image with a different noise value. Assuming the noise has zero mean, their network learns to remove the corruption. This setting implicitly assumes access to the distribution of uncorrupted images in order to generate different noisy versions of the same image, which is not our case.

2.3.3.2 Inpainting.

Image completion and inpainting have a long history. Methods developed in the early 2000 relied either on diffusion, by optimizing an energy function, e.g., (Ballester et al. 2000; Bertalmio et al. 2000) or on patch completion, e.g., (Simakov et al. 2008; Barnes et al. 2009). There is no learning at all involved for these methods. Hole filling is usually performed by using only local information from the image itself and not attempting to extract information from other images. There are some exceptions like (Hays et al. 2007) who made use of image databases for completing the holes in the target image. More recently, convolutional neural networks were used for image completion. Contrarily to the present work, they typically assume some form of supervision for mapping incomplete images onto reconstructed ones. Recent work usually relies on some form of adversarial training for obtaining sharp reconstructions. A reference work here is (Pathak et al. 2016) which uses content encoders in an image translation approach (Isola et al. 2016) to map masked images onto non-masked ones. This is one of the first works demonstrating that large holes could be filled by a learning approach. (Song et al. 2017) extended content encoders by using a refinement network in which a blurry initial hole-filling result is used as the input and then iteratively improved. Yu and al. (J. Yu et al. 2018) use a post-processing step based on contextual attention layers. In (J. Yu et al. 2018) and (Iizuka et al. 2017) adversarial training is performed at different levels of the image representation and discriminators are trained both on global -low frequency- and local -high frequency- features. Non-adversarial approaches have also been developed like e.g., (G. Liu et al. 2018) Liu et al. who introduce partial convolutions, where the convolution is masked and renormalized to be conditioned on valid pixel values only.

2.3.3.3 Unsupervised Inverse Problem

AmbientGAN (Bora et al. 2018) is an adversarial generative model aimed at generating images by being trained on noisy examples only and making use of a known stochastic measurement process as we do here. It does not perform completion but learns the distribution in the reconstruction space. Several works were later build on this model. MisGAN (Li et al. 2018) tackles the same problem as AmbientGAN (Bora et al. 2018) in the specific case of masked images like we do here. They extend the model in (Bora et al. 2018) by assuming that the mask distribution is unknown when it is supposed to be known for AmbientGAN. They train two generators, one as in (Bora et al. 2018) for generating an image, and one for learning the mask distribution. The latter is trained as a classical GAN since the mask is fully observed. They propose a conditional variant of this model dedicated to the imputation task requiring the training of an additional generator. Compared to our approach, they also learn

from incomplete data and do not rely on any supervision. Their assumption is even less restrictive than ours since they learn the nature of the noise itself. On the other hand, their model requires learning a generative model of the data prior to train a completion generator itself, when we address the problem more directly by training the imputer on the incomplete observations themselves.

CHAPTER 3

INCORPORATING PRIOR KNOWLEDGE IN FORECASTING SPATIO-TEMPORAL DATA

Contents

3.1 Introduction **28**
3.2 Physical Motivation **30**
3.2.1 Incorporating the Advection Diffusion Equation . . . 30
3.3 Model **31**
3.3.1 Model Description 31
3.3.2 Warping Scheme 33
3.3.3 Loss Function 34
3.4 Experiments **35**
3.4.1 Dataset 35
3.4.2 Experimental Setting 36
3.4.3 Model Architecture 37
3.4.4 Baselines 37
3.4.5 Results. 38
3.4.6 Quantitative Results 38
3.4.7 On the Generalization in Space and Time 41
3.5 Conclusion **43**
3.5.1 Additional Samples 43

Chapter abstract

In this chapter, we consider the use of Deep Learning methods for forecasting complex phenomena, like those occurring in natural physical processes. With the large amount of data gathered on these phenomena the data intensive paradigm could begin to challenge more traditional approaches elaborated over the years in fields like mathematics or physics. Using an example application, namely Sea Surface Temperature Prediction, we show how general background knowledge gained from the physics, under the form of Partial Differential Equation (PDE) could be used as a guideline for designing efficient Deep Learning models. In order to motivate the approach

and to assess its generality, we demonstrate a formal link between the solution of a class of differential equations underlying a large family of physical phenomena and the proposed model. Experiments and comparison with series of baselines including a state-of-the-art numerical approach is then provided.

The work in this chapter, made in collaboration with Emmanuel De Bezenac, has led to the publication of a conference paper:

- Emmanuel de Bezenac, Arthur Pajot, and Patrick Gallinari (Feb. 2018). "Deep Learning for Physical Processes: Incorporating Prior Scientific Knowledge". In: URL: https://openreview.net/forum?id=By4HsfWAZ

3.1 Introduction

A physical process is a sustained phenomenon marked by gradual changes through a series of states occurring in the physical world. Physicists and environmental scientists attempt to model these processes in a principled way through analytic descriptions of the scientist's prior knowledge of the underlying processes. Conservation laws, physical principles or phenomenological behaviors are generally formalized using differential equations. This physical paradigm has been, and still is the main framework for modeling complex natural phenomena like e.g., those involved in climate.

With the availability of very large datasets captured via different types of sensors, this physical modeling paradigm is being challenged by the statistical Machine Learning (ML) paradigm, which offers a prior-agnostic approach. However, despite impressive successes in a variety of domains as demonstrated by the deployment of Deep Learning (DL) methods in fields such as vision, language, speech, etc., the statistical approach is not yet ready to challenge the physical paradigm for modeling complex natural phenomena, or at least it has not demonstrated how to. We believe that knowledge and techniques accumulated for modeling physical processes in well-developed fields such as maths or physics could be useful as a guideline to design efficient learning systems and conversely, that the ML paradigm could open new directions for modeling such complex phenomena.

In this chapter we try to answer a fundamental question, that rose from those observations : how general knowledge gained from the physical modeling paradigm could help designing efficient ML models ?

We tackle these questions by considering a specific physical modeling problem: forecasting Sea Surface Temperature (SST). SST plays a significant role in analyzing and assessing the dynamics of weather and other natural systems. Accurately modeling and predicting such dynamics is critical in various ap-

plications such as weather forecasting, or planning of coastal activities. Since 1982, weather satellites have made huge quantities of very high resolution SST data available (Bernstein 1982). Standard physical methods for forecasting SST use coupled ocean-atmosphere prediction systems, based on the Navier Stokes equations. These models rely on multiple physical hypotheses and do not optimally exploit the information available in the data. On the other hand, despite the availability of large amounts of data, direct applications of ML methods do not lead to competitive state-of-the-art results, as will be seen in Section 3.4.

We use SST as a typical and representative problem of intermediate complexity. Our goal is not to offer one more solution to this problem, but to use it as an illustration for advancing on the challenges mentioned above. The way we handle this problem is general enough to be transferred to a more general class of transport problems.

We propose a Deep Neural Netwok model, inspired from general physical motivations which offers a new approach for solving this family of problems. We first motivate our approach by introducing in Section 3.2 the solution of a general class of Partial Differential Equation (PDE) which is a core component of a large family of transport and propagation phenomena in physics. This general solution is used as a guideline for introducing a Deep Learning architecture for SST prediction which is described in Section 3.3. Experiments and comparison with a series of baselines are introduced in Section 3.4.

The main contributions presented in this chapter are

- An example showing how to incorporate general physical background for designing a Neural Network (NN) aimed at modeling a relatively complex prediction task. We believe the approach to be general enough to be used for a family of transport problems obeying general advection-diffusion principles.

- Formal links between our model's prediction and the solution of a general advection diffusion PDE.

- An unsupervised model for estimating motion fields, given a sequence of images.

- A proof, on a relatively complex physical modeling problem, that full data intensive approaches based on deep architectures can be competitive with state of the art dedicated numerical method.

3.2 Physical Motivation

3.2.1 Incorporating the Advection Diffusion Equation

In this section, we introduce, the Advection Diffusion Equation, which is an example of PDE characterizing flow displacement. Forecasting consists in predicting future temperature maps using past records. Temperatures are acquired via satellite imagery. If we focus on a specific area, we can formulate the problem as prediction of future temperature images of this area using past images as:

$$I(x,t) = I(x + \Delta x, t + \Delta t). \tag{3.1}$$

Applying a first order Taylor expansion of the time and space in the right-hand side and moving the resulting terms to the left-hand side of equation Equation 3.1, we obtain the *advection equation*, also known as the Brightness Constancy Constraint Equation (BCCE):

$$\frac{\partial I}{\partial t} + (w.\nabla)I = 0, \tag{3.2}$$

Where ∇ denotes the gradient operator and w the motion vector $\frac{\Delta x}{\Delta t}$. This equation describes the temporal evolution of quantity I for displacement w. Note that this equation is also the basis for many variational methods for *Optical Flow*. To retrieve the motion, numerical schemes are applied, and the resulting system of equations, along with an additional constraint on w is solved for w. This motion can then be used to forecast the future value of I.

Advection alone is not sufficient to explain the evolution of many physical processes (including SST). *Diffusion* corresponds to the movement which spreads out the quantity I from areas of high concentration to areas of low concentration. Both advection and diffusion should be considered together. The following equation describes the transport of quantity I through advection and diffusion:

$$\frac{\partial I}{\partial t} + (w.\nabla)I = D\nabla^2 I. \tag{3.3}$$

∇^2 denotes the Laplacian operator and D the diffusion coefficient. Note that when $D \to 0$, we recover the advection Equation 3.2.
This equation describes a large family of physical processes (e.g., fluid dynamics, heat conduction, wind dynamics, etc.). Let us now state a result, characterizing the general solutions of Equation 3.3.

Theorem 3.1. [1] *For any initial condition $I_0 \in L^1(\mathbb{R}^2)$ with $I_0(\pm\infty) = 0$, there exists a unique global solution $I(x,t)$ to the advection-diffusion equation Equation 3.3:*

$$I(x,t) = \int_{\mathbb{R}^2} k(\, x - w,\, y)\; I_0(y)\, dy, \tag{3.4}$$

Where $k(u,v) = \frac{1}{4\pi Dt} e^{-\frac{1}{4Dt}\|u-v\|^2}$ is a radial basis function kernel, or alternatively, a 2 dimensional Gaussian probability density with mean u and variance $2Dt$.

For this theorem, we make the hypothesis that w is constant locally (around x) in space and time.

Equation 3.4 provides a principled way to calculate $I(x,t)$ for any time t using the initial condition I_0, provided the motion w and the diffusion coefficient D are known. It states that quantity $I(x,t)$ can be computed from the initial condition I_0 via a convolution with a Gaussian probability density function. In other words, if I was used as a initial condition for the evolution of the SST and the surface's underlying advecting mechanisms were known, future surface temperatures could be predicted from previous ones. Unfortunately neither the initial conditions, the motion vector nor the diffusion coefficient are known.

They have to be estimated from the data. Inspired from the general form of Equation 3.4, we propose a method, expressed as a Deep Learning architecture for predicting the SST evolution. This model will learn to predict a motion field analog to the w in Equation 3.4, which will be used to predict future images. This method can then be see as a warping of previous acquisitions along motion w.

3.3 Model

3.3.1 Model Description

The model consists of two main components, as illustrated in Figure 3.1. One predicts the motion field from a sequence of past input images and the other warps the last input image using the motion field from the first component, in order to produce an image forecast. The entire system is trained in an end-to-end fashion, using only the supervision from the target SST image. By doing so, we are able to produce an interpretable latent state which corresponds in our problem to the velocity field advecting the SST.

Let us first introduce some notations. Each SST image I_t is acquired on a bounded rectangle of $\mathbb{R}^2$, named Ω. We denote $I_t(x)$ and $w_t(x)$ the sea surface temperature and the two-dimensional motion vector at time $t \in \mathbb{R}$ at position

1. A proof of this theorem is available in Section 7.2

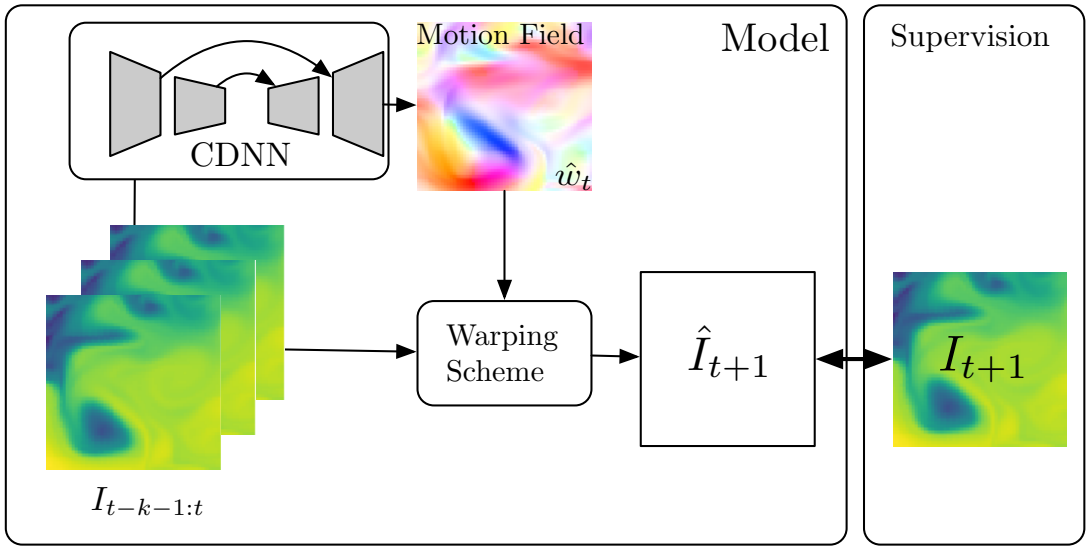


Figure 3.1 – Motion is estimated from the input images ($I_{t-k-1:t}$) with a Neural Network. A warping scheme then displaces the last input image along this motion estimate to produce the future image. The error signal is calculated using the target future image I_{t+1}, and is backprogated through the warping scheme to correct the Neural Network. To produce multiple time-step forecasts, the predicted image is fed back in the Neural Network in an autoregressive manner.

$x \in \Omega$. $I_t : \Omega \to \mathbb{R}$ and $w_t : \Omega \to \mathbb{R}^2$ represent the temperatures and the motion vector field at time t defined on Ω. When time t and position x are available from the context, we will drop the subscript t from $w_t(x)$ and $I_t(x)$, along with x for clarity. Given a sequence of k consecutive SST images $\{I_{t-k-1}, ..., I_t\}$ (also denoted as $I_{t-k-1:t}$), our goal is to predict the next image I_{t+1}.

As indicated in Section 3.2.1, provided the underlying motion field is known, one can compute SST forecasts. Let us introduce how the motion field is estimated in our architecture. We are looking for a vector field w which when applied to the geometric space Ω renders I_t close to I_{t+1}, i.e. $I_{t+1}(x) \simeq I_t(x + w(x))$, $\forall x \in \Omega$. During inference, if I_{t+1} were known, we could estimate w, but I_{t+1} is precisely what we are looking for. Instead, we choose to use a NN architecture to predict a motion vector for each pixel.

Generally, and this is the case for our problem, we do not have a direct supervision on the motion vector field, since the target motion is usually not available. Using the warping scheme introduced below, we will nonetheless be able to supervise w, based on the discrepancy of the warped version of the I_t image and the target image I_{t+1}.

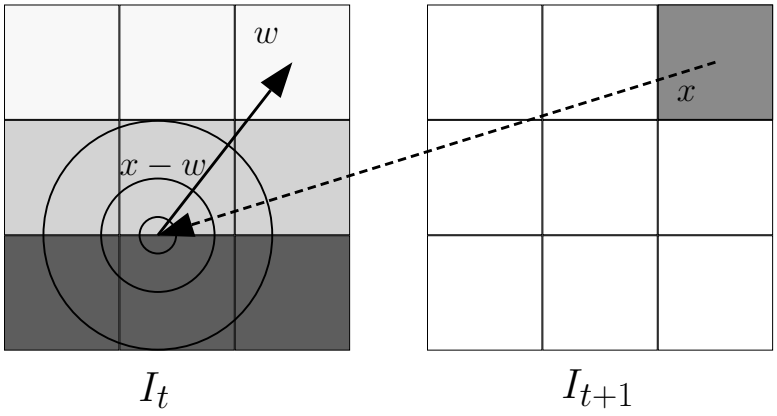


Figure 3.2 – Warping scheme. To calculate the pixel value for time $t+1$ at position x, we first compute its previous position at time t, *i.e.* $x - w$. We then center a Gaussian in that position in order to obtain a weight value for each pixel in I_t based on its distance with $x - w$, and compute a weighted average of the pixel values of I_t. This corresponds to the diffusion mechanism. This weighted average will correspond to the new pixel value at x in I_{t+1}.

3.3.2 Warping Scheme

Discretizing the solution of the advection-diffusion equation in Section 3.2.1 by replacing the integral with a sum, and setting image I_t as the initial condition, we obtain a method to calculate the future image, based on the motion field estimate $\hat{w}$. The latter is used as a warping scheme:

$$\hat{I}_{t+1}(x) = \sum_{y \in \Omega} k(\, x - \hat{w}(x),\, y)\; I_t(y) \tag{3.5}$$

Where $k(x - \hat{w}, y) = \frac{1}{4\pi D \Delta t} e^{-\frac{1}{4D\Delta t}\|x - \hat{w} - y\|^2}$ is a radial basis function kernel, as in Equation 3.4, parameterized by the diffusion coefficient D and the time step value Δt between t and $t+1$ and $\hat{w}$ is the estimated value of the vector flow w. To calculate the temperature for time $t+1$ at position x, we compute the scalar product between $k(x - \hat{w}, .)$, a Gaussian centered in $x - \hat{w}$, and the previous image I_t. Simply put, it is a weighted average of the temperatures I_t, where the weight values are larger when the pixel's positions that are closer to $x - \hat{w}$. Informally, $x - \hat{w}$ corresponds to the pixel's previous position at time t. See Figure 3.2.

Equation 3.5 simply says that estimate $\hat{I}_{t+1}(x)$ is the result of a convolution of the whole image I_t with the kernel $k(\, x - \hat{w}(x),\, y)$ where y describes the domain Ω of I_t.

As seen by the relation with the solution of the advection-diffusion equation, the proposed warping mechanism is then clearly adapted to the modeling of phenomena governed by the advection-diffusion equation. Fluid forecasting is a particular case, but the proposed scheme can be used for any problems in which advection and diffusion are occurring. Moreover, this warping scheme is entirely differentiable, allowing backpropagation of the error signal to the motion field estimating module.

This warping mechanism has been inspired by the model presented in (Jaderberg et al. 2015), originally designed to be incorporated as a layer in a convolutional neural network architecture in order to gain invariance under geometric transformations. Using the notations in (Jaderberg et al. 2015), when the inverse geometric transformation $\mathcal{T}_\theta$ of the grid generator step is set to $\mathcal{T}_\theta(x) = x - \hat{w}(x)$, and the kernels $k(\,.\,;\Phi_x)$ and $k(\,.\,;\Phi_y)$ in the sampling step are radial basis function kernels, we recover our warping scheme. The latter can be seen as a specific case of the Spatial Transformer Network (STN), without the localization step. It theoretically grounds the use of the STN for Optical Flow in many recent articles (Patraucean et al. 2015; Finn et al. 2016): in Equation 3.3, when $D \to 0$, we recover the Brightness Constancy Constraint Equation, used in the latter.

3.3.3 Loss Function

At each iteration, the model aims at forecasting the next observation, given the previous ones. We evaluate the discrepancy between the warped image $\hat{I}_{t+1}$ and the target image I_{t+1} using the Charbonnier penalty function $\rho(x) = (x + \epsilon)^{\frac{1}{\alpha}}$, where ϵ and α are parameters to be set. Note that with $\epsilon = 0$ and $\alpha = \frac{1}{2}$, we recover the ℓ_2 loss.

The Charbonnier penalty function is known to reduce the influence of outliers compared to an l_2 norm. It behaved slightly better in preliminary experiments. We have also tested the Laplacian pyramid loss (Ling et al. 2016), where we enforce convolutions of all deconvolutional layers to be close to down-sampled versions of the target image in the Charbonnier penalty sense, but we have observed an overall decrease in generalization performance.

The proposed neural network model has been designed according to the intuition gained from general background knowledge of a physical phenomenon, here advection-diffusion equations. Additional prior knowledge – expressed as partial differential equations, or through constraints – can be easily incorporated in our model, by adding penalty terms in the loss function. As the displacement w is explicitly part of our model, one strength of our model is its capacity to apply some regularization term directly on the motion field. The following quantities are prior knowledge that could be seen as constraints. In our experiments, we tested the influence of different terms: divergence $\nabla . w_t(x)^2$ which

locally control the variation of the motion field, magnitude $\|w_t(x)\|^2$ which controls the amplitude of the motion field, and smoothness $\|\nabla w_t(x)\|^2$ which controls the amplitude of its variation. They are, in our case, hyperparameters set by cross validation. The loss function can be written as :

$$\begin{aligned}\mathcal{L}_t = &\sum_{x\in\Omega} \rho(\hat{I}_{t+1}(x) - I_{t+1}(x)) \\ &+ \lambda_{\text{div}}(\nabla . w_t(x))^2 + \lambda_{\text{magn}} \|w_t(x)\|^2 + \lambda_{\text{grad}} \|\nabla w_t(x)\|^2 .\end{aligned} \tag{3.6}$$

3.4 Experiments

In this section we evaluate our model, both quantitatively and qualitatively. We consider a dataset of medium complexity representing the evolution of the sea surface temperature. We evaluate our method with respect to its ability to predict observations and to reproduce the dynamics of the hidden state. For the first two datasets, we use the full initial condition as input. For the last dataset, we only have access to a subset of the states and weight propose a variant of our approach in order to accommodate this situation.

3.4.1 Dataset

Since 1982, high resolution SST data has been made available by the *NOAA6* weather satellite (Bernstein 1982). Dealing directly with these data requires a lot of preprocessing (e.g., some regions are not available due to clouds hindering temperature acquisition). In order to avoid such complications which are beyond the scope of this work, we used synthetic but realistic SST data of the Atlantic Ocean generated by a sophisticated simulation engine: NEMO (Nucleus for European Modeling of the Ocean) engine Madec 2008. NEMO is a state-of-the-art modeling framework of ocean-related engines. It is a primitive equation model adapted to the regional and global ocean circulation problems. Historical data is accumulated in the model to generate a synthesized estimate of the states of the system using data *reanalysis*, a specific data assimilation scheme, which means that the data does follow the true temperatures. The resulting dataset is built of daily temperature acquisitions of 481 by 781 pixels, from 2006-12-28 to 2017-04-05 (3734 acquisitions).

We extract 64 by 64 pixel sized sub-regions as indicated in Figure 3.3.

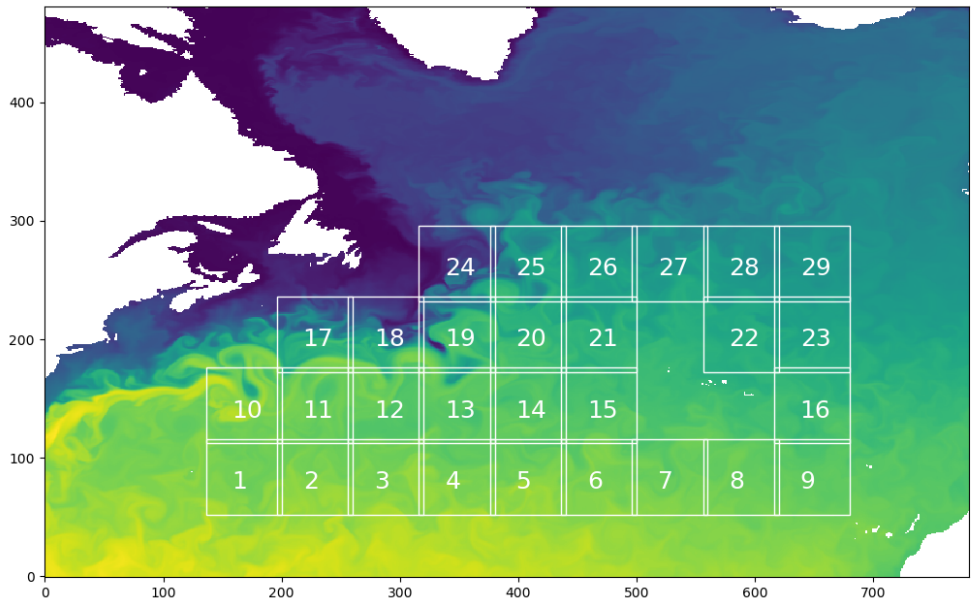


Figure 3.3 – Sub regions extracted for the dataset. Test regions are regions 17 to 20.

3.4.2 Experimental Setting

We decompose the simulations into training sequences of fixed length, using 4 time steps as input, and 6 time steps for the target sequence. More precisely, we have $I_{t:t+4}$ as an input of the model. We then estimate I_{t+5} and concatenate it to the previous input in order to estimate the following images until I_{t+10}. The loss is computed between $I_{t+4:t+10}$ and the estimated $\hat{I}_{t+4:t+10}$. In practice, the cost functional $\mathcal{L}$ is estimated on a minibatch of sequences from the dataset and optimized using stochastic gradient descent. We use data from years 2006 to 2015 for training and validation (94743 training examples), and years 2016 to 2017 for testing. We withhold 20% of the training data for validation, selected uniformly at random at the beginning of each experiment. For the tests we used sub-regions enumerated 17 to 20 in Figure 3.3, where the interactions between hot and cold waters make the dynamics interesting to study. All the regions numbered in Figure 3.3, from 2006 to 2015 were used for training [2]. Each sequence of images used for training or for evaluation corresponds to a specific numbered sub-region. We make the simplifying hypothesis that the data in a single sub-region contains enough information to forecast the future of the sub-region. As the forecast is for a small temporal horizon, we can assume that the influence from outside the region is small enough. The boundary conditions

2. non-numbered regions correspond to land and not sea on the figure

are set to zero, which means that any pixel dispaced inside of each box, will be set to zero.

Concerning the constraints on the vector field w (Equation 3.6. the regularization coefficients selected via validation are $\lambda_{\text{div}} = 1$, $\lambda_{\text{magn}} = 0$ and $\lambda_{\text{grad}} = 0.4$. The diffusion coefficient D was set to 0.45 by cross validation. We also compare the results with the model without any regularization.

3.4.3 Model Architecture

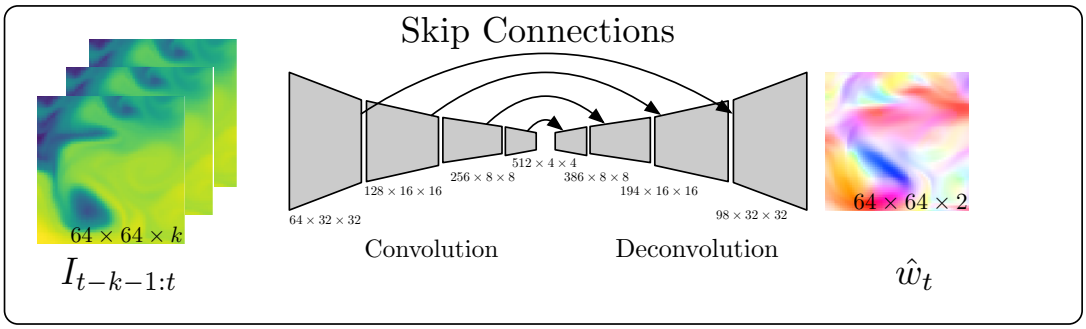


Figure 3.4 – Architecture of the NN motion estimation component. For the estimated motion flow $\hat{w}_t$, colors correspond to the flow orientation and color intensity to the flow intensity

As shown in Figure 3.4, the network that we used during our experiments, makes use of skip connections (He et al. 2015), allowing fine-grained information from the first layers to flow through in a more direct manner. We use a *Batch Normalization* layer between each convolution, and *Leaky ReLU* (with parameter value set to 0.1) non-linearity between convolutions and transposed-convolutions. We used $k = 4$ concatenated images $I_{t-k-1:t}$ as input for training. We have selected this architecture experimentally, testing different state-of-the-art convolution-deconvolution network architectures.

3.4.4 Baselines

We compare our model, which is called Prior Knowledge Network (PKN), with several baselines. Each model is evaluated with a mean square error metric, forecasting images on a horizon of 6 (we forecast from I_{t+1} to I_{t+6} and then average the Mean Squared Error (MSE)). The hyperparameters are tuned using the validation set. Neural network based models are run on a Titan Xp GPU, and runtime is given for comparison purpose.

Our reference model for forecasting is (Bérériat et al. 2015), a numerical assimilation model which relies on data assimilation. In (Bérériat et al. 2015), the ocean's dynamics are modeled using shallow water equations (Vallis 2017) and the initial conditions, along with other terms, are estimated using complex data assimilation techniques (Trémolet 2006). This is a state-of-the-art assimilation model for predicting ocean dynamic, here SST.

The other baselines are :

- An autoregressive convolutional-deconvolutional neural network (ACDNN), with an architecture similar to the Neural Network module described in Section 3.4.3, but trained to predict the future image directly, without explicitly representing the motion vector field. Each past observation is used as an input channel (the 4 input images used in the experiments are concatenated), and the output is used as new input for multi-step forecasting, as described in Section 3.4.2.
- (X. Shi et al. 2015), a recurrent model similar to Long-Short Term Memory (LSTM), which uses convolutional transitions in the inner LSTM module. This model is described in Section 2.1.3.
- The model in (Mathieu et al. 2015) which is a multi-scale ACDNN trained as a Generative Adversarial Network (GAN). This model is described in Section 2.1.3.

3.4.5 Results.

3.4.6 Quantitative Results

Model	Average Score (MSE)	Average Time
Numerical model (Bérériat et al. 2015)	1.99	4.8 s
ConvLSTM (X. Shi et al. 2015)	5.76	0.018 s
ACDNN	15.84	0.54 s
GAN Video Generation (Mathieu et al. 2015)	4.73	0.096 s
PKN with regularization	**1.42**	0.040 s
PKN without regularization	2.01	0.040 s

Table 3.1 – Average score and average time on test data. Average score is calculated using the *mean square error* metric (MSE), time is in seconds.

Quantitatively, our model performs well. The MSE score is better than any of the baselines. The closest neural network baseline is described in (Mathieu

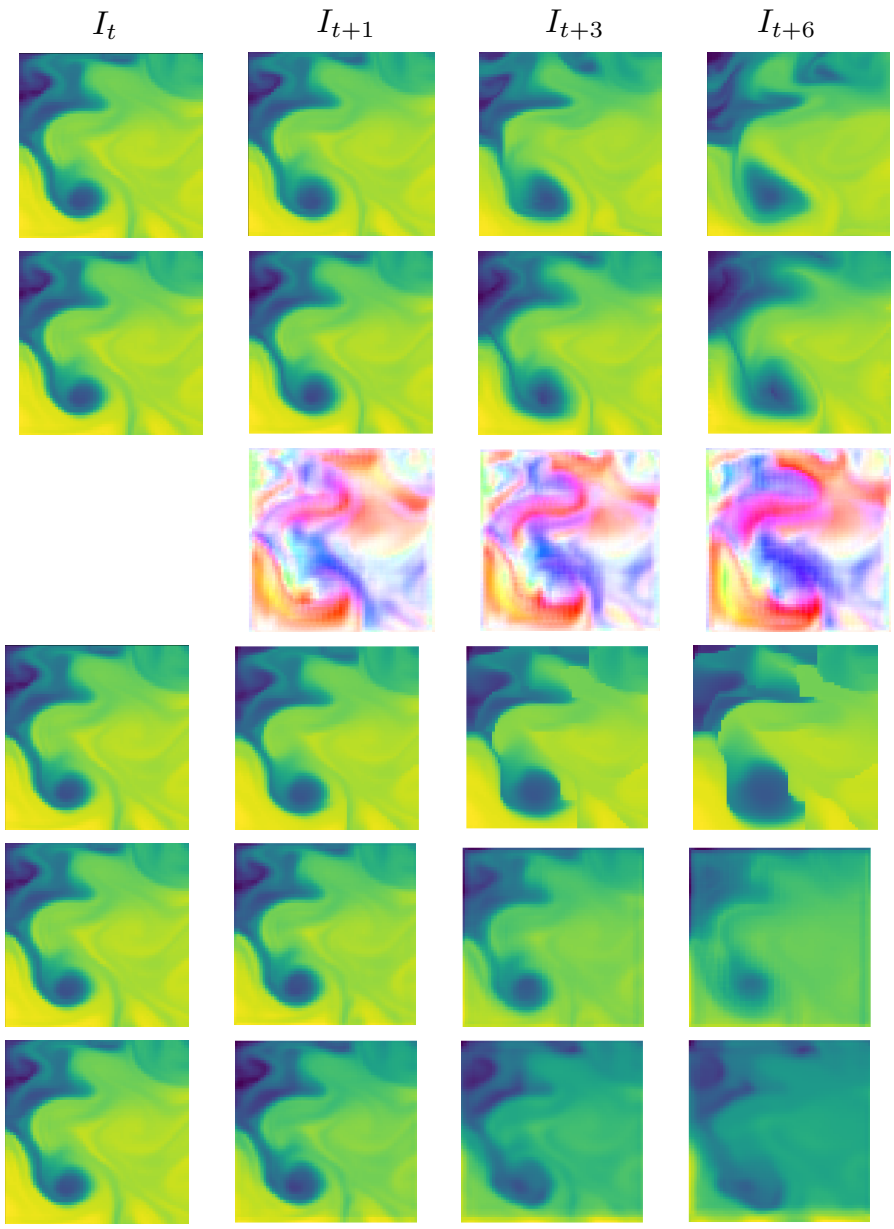


Figure 3.5 – From top to bottom: target, our model prediction, our model flow, numerical assimilation model, ACDNN, *ConvLSTM*. Data correspond to daily temperatures from January 17 to January 23, 2017, for regian 19.

et al. 2015) and regularizes a regression convolution-deconvolution model with a GAN. The performance is, however, clearly below the proposed model and it does not allow to easily incorporate prior constraints inspired from the physics of the phenomenon. ACDNN is a direct predictor of the image sequence, implemented via a NN module identical to the one used in our model. Its performance is poor. Clearly, a straightforward use of prediction models is not adapted to the complexity of the phenomenon. The Convolutional Long-Short Term Memory (ConvLSTM) performs better: as opposed to the ACDNN, it seems to be able to capture a dynamic, although not very accurately. Overall, direct prediction models are not able to capture the complex underlying dynamics and they produce blurry sequences of images. The GAN explicitly forces the network output to eliminate the blurring effect and then makes it able to capture short term dynamics. The state-of-the-art numerical model (Bérériat et al. 2015), performs well and has comparable performance with PKN, although it incorporates more prior constraints. This shows that pure ML models, when conceived adequately and when trained with enough data, can be competitive with state-of-the-art dedicated models. Regularizing the motion vector w notably increases the performance with respect to the unregularized model.

As for the running time, the proposed model is extremely fast, being just above the ConvLSTM model of (X. Shi et al. 2015). The running time of (Bérériat et al. 2015)'s model is not comparable to the others. It was run on a CPU (no GPU code) when all the others were run on Titan Xp GPUs. However, an optimization procedure is required to estimate the motion field, and it is clearly slower than the straightforward NN predictions. Moreover, in order to prevent the numerical scheme from diverging, multiple intermediate forecasts are required.

Besides MSE, we need to analyze the prediction samples qualitatively. Figure 3.5 shows predictions obtained by the different models. On the top row of Figure 3.5, the ground truth for a sequence of 4 temperature images corresponding to time t, $t+1$, $t+3$ and $t+6$. The second row corresponds to our regularized model prediction at times $t+1$, $t+3$ and $t+6$ (time t corresponds to the last input image, it is repeated on each row). The prediction is close to the target for $t+1$, $t+3$ and starts to move away at time $t+6$. The third row shows the motion flow estimated by the model. Each color in the flow images corresponds to a motion vector (see Figure 3.6). There is clearly a strong evolving dynamic captured for this sequence. Row 4 is the numerical assimilation model of (Bérériat et al. 2015). It also clearly captures some dynamics and shows interesting patterns, but it tends to diverge when the prediction horizon increases. The ACDNN model (row 5) rapidly produces blurry images; it does not preserve the temperatures and does not seem to capture any dynamics. On row 6 are plotted the predictions of the ConvLSTM model. Temperature is not

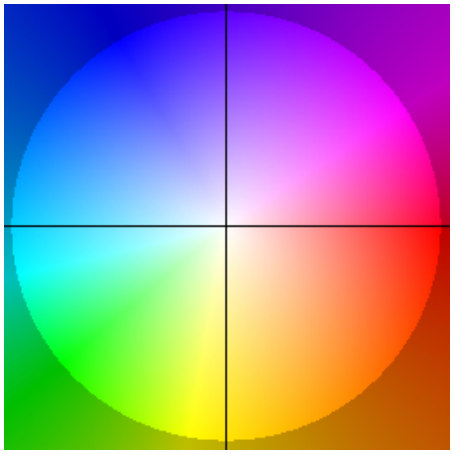


Figure 3.6

preserved, as it seems to fade away, and although a dynamic is captured, it does not correspond to the target. Overall, the proposed model seems to forecast SST quite accurately, while retrieving a coherent motion vector field. Additional samples are available at Section 3.5.1.

3.4.7 On the Generalization in Space and Time

The ability of the model to adapt to other conditions should be evaluated on other regions. We present below complementary experiments aimed at assessing the potential of the proposed model for forecasting SST on sequences distant in time and space from the ones used for training.

3.4.7.1 Temporal Dimension

In Section 3.4, training has been performed on data from 2006 - 2015 and testing on the period 2016-2017. In order to provide some indication of the model behavior on more distant time intervals between train and test data, we have performed experiments using the same regions (17 to 20) as in Section 3.4, but using the period 2011 to 2017 for training and period 2006 to 2010 for testing. Figure 3.7 shows the MSE curve on this test set, each point corresponding to the mean MSE on predictions performed on 6 days ahead the current date. The most important conclusion is probably that the MSE error remains in the same range for all these years. All the yearly error curve shows a clear seasonal phenomenon with a higher prediction error during summer. A similar behavior has been observed when exchanging train and test data.

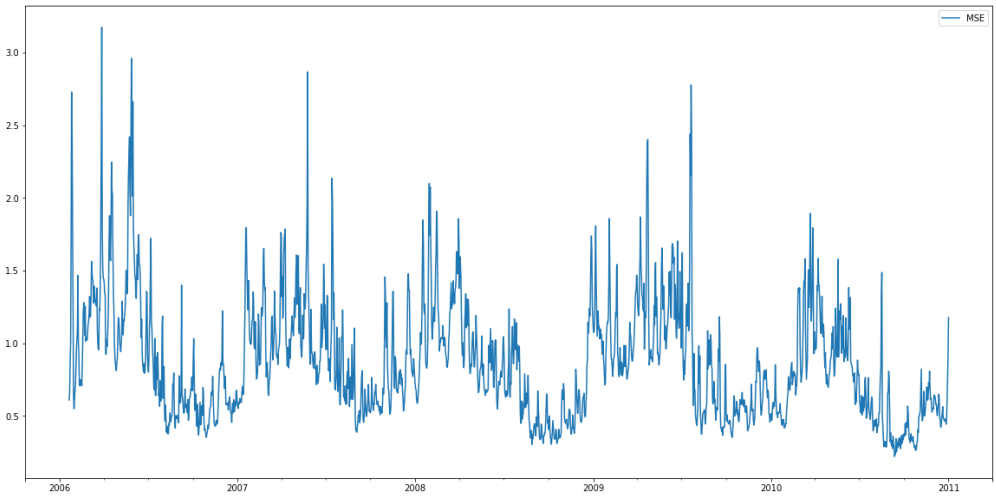


Figure 3.7 – Evaluation of our model's accuracy in time on data from 2006 to 2010 using data from 2011 to 2017 for training. Regions 17 to 20 were used for both periods. Each day, we produce daily forecasts for 6 days ahead and calculate the associated mean square error. The color of the flow represents the direction of the direction each of the vector as illustrated in Figure 3.6.

3.4.7.2 Spatial Dimension

In the experiments, the models have been trained and evaluated on selected regions (numbered 17 to 20 in Figure 3.3), considered as the most interesting for the observed dynamics.

	Test Regions 17 & 18	Test Regions 8 & 9
Model trained on Regions 17 & 18	1.43	1.22
Model trained on Regions 8 & 9	1.90	1.19

Table 3.2 – Evaluation of our model's spatial generalization ability. We train our model on two distinct regions and calculate the MSE on both regions for each trained model.

We describe below some results providing indications on how the model performs on regions different from the training ones. For these experiments, the model has been trained on regions 17 and 18 in Figure 3.3 and tested on two other regions (regions 8 and 9), and vice versa (trained on 8 and 9 and tested on 17 and 18). The two couples of regions have been selected so as to have different latitude and longitude coordinates. The underlying physical processes generating the data are known to be different in these regions: the overall motion in regions 17 and 18 is greater, and the difference between extreme temperatures is larger, compared to regions 8 and 9. Experimental conditions

are similar to the one described in section Section 3.4.2, i.e. 2006-2015 have been used for training and 2016-1017 for testing.

Results in Table 3.2 show that the model generalizes reasonably well to unseen data from distant spatial regions, with a slight decrease in performance when training and test regions do not correspond. The performance loss is 0.47 for regions (17, 18) which show a strong dynamic, whereas it is only 0.03 for regions (8, 9) for which the dynamics are more stable. Most notably, MSE performance depends more on the region itself than on the train/ test conditions. Error is always higher in regions with strong dynamics (17, 18) than on more stable regions (8, 9) whatever the train/ test conditions are. Note that to further improve the results on distant data, it is possible to fine-tune the model using data from the studied regions.

3.5 Conclusion

By using as an example application a relatively complex problem concerning ocean dynamics, we proposed a principled way to design Deep Learning models using inspiration from the physics. The proposed approach can be easily generalized to a class of problems for which the underlying dynamics follow advection-diffusion principles. We have compared the proposed approach to a series of baselines. It is able to reach performance comparable to a state-of-the-art numerical model and clearly outperform alternative NN models used as baselines.

3.5.1 Additional Samples

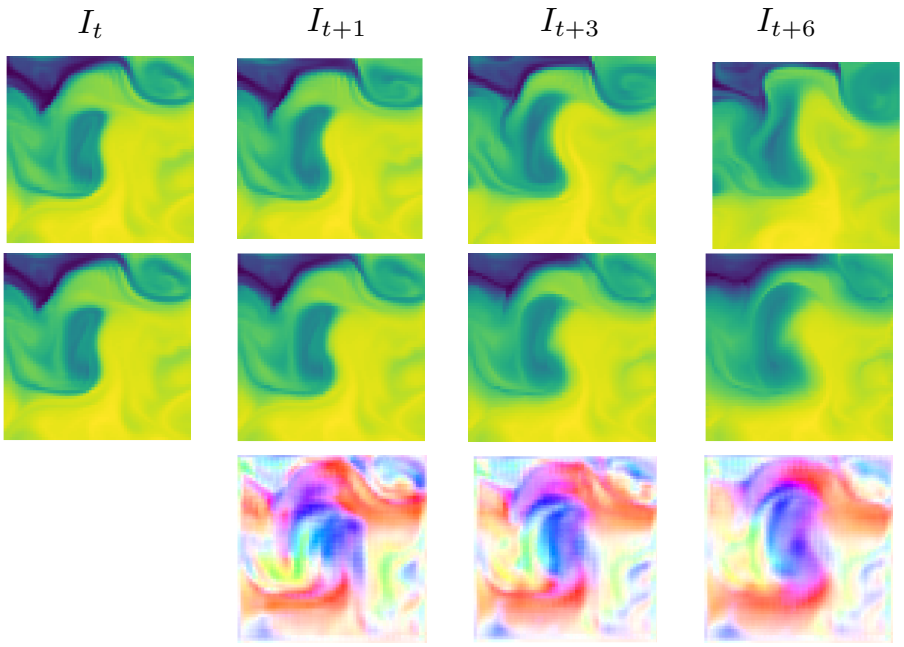


Figure 3.8 – Output for the 6 of May to the 9 of May 2016, Output , From top to bottom: target, our model prediction, our model flow. The color of the flow represents the direction of each of the vector as illustrated in Figure 3.6.

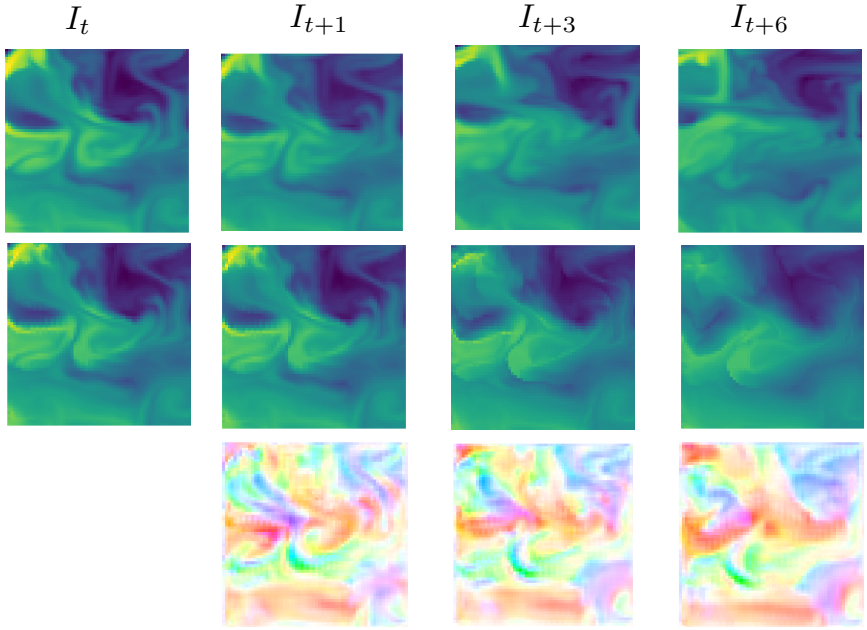


Figure 3.9 – Output for the 6 of January to the 9 of January 2016. From top to bottom: target, our model prediction, our model flow. The color of the flow represents the direction of each of the vector as illustrated in Figure 3.6.

CHAPTER 4

LEARNING DYNAMICAL SYSTEMS FROM PARTIAL OBSERVATIONS

Contents

4.1 Introduction 48
4.2 Methodology 49
4.2.1 Our Approach 49
4.2.2 Models 53
4.3 Experiments 53
4.3.1 Datasets. 54
4.3.2 Experimental setting. 54
4.3.3 Results. 56
4.4 Conclusion 61

Chapter abstract

We consider the problem of learning the dynamics of physical processes evolving in space and in time, given only partial observations of the state. We propose a natural data-driven framework, where the system's dynamics are modeled by an unknown time-varying differential equation, and the evolution term is estimated from the data, using a neural network. Given an initial state, an ODE solver can then be used to compute any future state. We qualitatively and quantitatively study the results of our method over simulations of fluid equations. We show that our method not only successfully forecasts future observations, consistently outperforming classical baselines, but it also learns to closely reproduce the unobserved dynamics of the state without direct supervision on the latter when the true initial state is given as input. We also show that our method can still be successfully applied when the initial state is not available and that it produces an interpretable hidden state even in this case.

The work in this chapter was made in collaboration with Emmanuel De Bezenac and Ibrahim Ayed. It has led to a paper:

- Ibrahim Ayed, Emmanuel de Bézenac, Arthur Pajot, Julien Brajard, and Patrick Gallinari (Feb. 2019). "Learning Dynamical Systems from Partial Observations". In: *arXiv:1902.11136 [physics]*. URL: http://arxiv.org/abs/1902.11136

4.1 Introduction

Let us consider a dynamical system to describe a real-world physical process. The *state* X_t ($t \in \mathbb{R}_+$ stands for time) of the dynamical system can then be defined as a vector-valued function on a space $x \in \Omega$:

$$\forall t, \ \frac{dX_t}{dt} = F(X_t). \tag{4.1}$$

In this equation, X_t is sufficient to describe the temporal dynamics of the underlying process. For example, in the incompressible Navier Stokes equations a state can be the concatenation of the density and velocity fields of the fluid as those are enough for describing the evolution of the system, while pressure, an additional variable of interest, can be computed from those variables.

With the availability of very large amounts of data captured via diverse sensors and recent advances of statistical methods, a new data-driven paradigm for modeling dynamical systems is emerging, where relations between the states are no longer handcrafted, but automatically discovered based on the available observations. This problem can be approached by considering some class of admissible functions $\{F_\theta\}$, and looking for a θ such that the solution X^θ of:

$$\frac{dX_t}{dt} = F_\theta(X_t) \tag{4.2}$$

fits the measured data.

In this chapter, we consider the problem of learning complex spatiotemporal dynamical systems with neural networks from observation, which are only partially informative with respect to the full state.

First, we formulate the problem as a continuous-time optimization problem under the constraints of a Partial Differential Equation (PDE), where the parameters of the neural network are viewed as optimized variables. From this, we then present a natural algorithm for solving the resulting optimization problem, placing the neural network in an Ordinary Differential Equation solver in order to produce future predictions. Finally, we successfully apply our method to three increasingly challenging datasets and show promising results, comparing our approach to standard baselines.

- We propose a general model, parametrized with neural networks, which learns a state representation and its dynamics given partial observations.

- We present experiments on the Navier-Stokes equations exploring the properties of our model in each setting.

4.2 Methodology

In this section, we present the optimization problem defining our model to learn partially observed dynamics as well as the training algorithm we used.

4.2.1 Our Approach

4.2.1.1 Continuous State Space Models

We consider space-time dynamics for which X can be written as a function of $(t, x) \in \mathbb{R}_+ \times \Omega$ where t and x are respectively the time and space variables, $\Omega \subset \mathbb{R}^d$ is the domain over which we study the system. The spatial vector-valued function X_t contains the quantities of interest describing a studied physical system at time t.

In a realistic setting, the state is generally only partially observed *e.g.*, when studying the ocean's circulation, variables contained in the system's state such as temperature or salinity are observable, while others such as velocity or pressure are not. In other words, the measured data is only a projection of the complete state X_t. This measurement process is modeled here with a fixed operator $\mathcal{H}$ linking the system's state X_t to the corresponding observation Y_t:

$$Y_t = \mathcal{H}(X_t)$$

In the following, $\mathcal{H}$ is supposed to be known, fixed and differentiable. In most practical cases, this hypothesis is verified as $\mathcal{H}$ can usually be represented as a smooth operator. Let us note that, generally, the measurement process represents a considerable loss of information compared to the case where X is available, as the measurements may be sparse and low-dimensional.

Moreover, we assume that X obeys a differential equation of the general form of equation 4.1, with an initial condition X_0. This leads us to the following continuous state space model:

$$\begin{cases} X_0 \\ \dfrac{dX_t}{dt} = F(X_t) \\ Y_t = \mathcal{H}(X_t) \end{cases} \tag{4.3}$$

4.2.1.2 Optimization Problem

Our goal is to learn the differential equation driving the dynamics of a smooth state function X for which we only have supervision over observations Y through a fixed operator $\mathcal{H}$. In order to ensure that our dynamical system at least explains the observations, we define a cost functional of the form:

$$\mathcal{J}(Y, \widetilde{Y}) = \int_0^T \|Y_t - \widetilde{Y}_t\|^2 \mathrm{dt} \tag{4.4}$$

Here, Y is a spatiotemporal field representing observations of the studied system, $\widetilde{Y}$ is the output of the system, and $\|\cdot\|$ the L2 norm.

The state X_t is constrained to follow the dynamics described by equation 4.2, starting from an initial condition X_0. The optimization problem is now formulated as :

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & \mathbb{E}_{Y \in \text{Dataset}} \left[\mathcal{J}(Y, \mathcal{H}(X)) \right] \\ \text{subject to} \quad & \frac{dX_t}{dt} = F_\theta(X_t), \\ & X_0 = g_\theta(Y_{-k}, \check{X}_0) \end{aligned} \tag{4.5}$$

where F_θ is a smooth vector valued function defining the trajectory of X, and g_θ gives us the initial condition X_0. In other words, θ parameterizes both the dynamics through F and the initialization through g. In particular, if a full initial state is given as input to the system, g_θ can be taken as independent from any parameter and does not need to be learned.

For any θ, we assume that F and g are such that there always exists a solution to the equation :

$$\begin{cases} X_0 = g_\theta(Y_{-k}, \check{X}_0) \\ \dfrac{dX_t}{dt} = F_\theta(X_t) \end{cases} \tag{4.6}$$

In the following, we will call such a solution X^θ.

4.2.1.3 Adjoint State Method

In order to construct a gradient descent algorithm to solve the problem 4.5, we need to find the gradient of the cost functional under the given constraints, *i.e.* the differential of $\theta \to \mathbb{E}_Y \mathcal{J}(Y, \mathcal{H}(X^\theta))$ (**plessix2006review**). However, this implies calculating $\dfrac{\partial X^\theta}{\partial \theta}$, which is often very computationally demanding, as it implies solving $\dim(\theta)$ forward equations. The adjoint state method avoids those costly computations by considering the Lagrangian formulation of the constrained optimization problem. Adjoint State Method is a technique allowing

to solve a constraint optimization problem well adapted when the number of parameters to optimize is large. We introduce below a method for the case of continuous systems, constrained by an Ordinary Differential Equation (ODE).

Theorem 4.1 (Adjoint State Equation)**.** *The gradiend of $\mathcal{J}$ w.r.t parameters θ, for a solution X^θ to the initial value problem discribed in 4.5 can calculated as follows.*

$$\frac{\partial}{\partial\theta}\mathcal{J}(Y,\mathcal{H}(X^\theta)) = -\int_0^T \left\langle \lambda_t, \partial_\theta F_\theta(X_t^\theta)\right\rangle \mathrm{dt} - \langle \lambda_0, \partial_\theta g_\theta\rangle \tag{4.7}$$

where λ is the solution of:

$$\partial_t\lambda_t = A_t\lambda_t + B_t \tag{4.8}$$

solved backwards, starting with $\lambda_T = 0$, and where :

$$A_t = -(\partial_X F_\theta(X_t^\theta))^\star$$

and

$$B_t = 2(\partial_X\mathcal{H}(X_t^\theta))^\star(\mathcal{H}(X_t^\theta) - Y_t)$$

where $M^\star$ denotes the adjoint operator of linear operator M.

Here A is the adjoint of F The proof can be found in Section 7.3. When training, with this result, for a given value of θ, we can solve the forward equation 4.2 to find X^θ. Then, λ can be solved backwards as its equation only depends on X^θ which gives us all necessary elements to calculate the gradient of $\mathcal{J}$. This gives us the following iterative algorithm to solve the optimization problem, starting from a random initialization of θ :

1. Solve the forward state equation 4.6 to find X^θ ;
2. Solve the backward adjoint equation 4.8 to find the corresponding λ (see Theorem 4.1);
3. Update θ in the steepest descent direction using equation 4.7.

From these steps (and taking into account the estimation of the initial state, further explained in Section 4.3), we now have a general algorithm for training. *At inference*, we use the learned $g_{\theta^\star}$ to compute an initial state then simply solve the forward equation with the learned [1]

There are many possible choices regarding the way the different equations are solved in practice: Those are discussed in Section 4.2.1.4.

1. In particular, it is important to note that no further updates or corrections are made and no additional observations are needed.

Algorithm 4.1 Training Procedure

Input: Training samples $\{(Y_{-k}, \breve{X}_0,), Y_{+l}\}$.
Guess initial parameters θ
while not converged **do**
  Randomly select sample sequence $\{(Y_{-k}, \breve{X}_0,), Y_{+l}\}$
  if Initial State is Fully Observed **then**
    $X_0 \leftarrow \breve{X}_0$
  else
    $X_0 \leftarrow g_\theta(Y_{-k}, \breve{X}_0)$
  end if
  Solve Forward $\frac{dX_t}{dt} = F_\theta(X_t),\ X(0) = X_0,\ t \in [0, l]$
  Solve Backward $\frac{d\lambda_t}{dt} = A_t\lambda_t + B_t,\ \lambda_l = 0,\ t \in [0, l]$
  Compute gradient $\frac{\partial \mathcal{J}}{\partial \theta}(X^\theta)$
  Update θ in the steepest descent direction
end while
Output: Learned parameters θ.

4.2.1.4 Approximate Solutions

While our algorithm seems straightforward, solving the forward and backward equations (4.2, 4.8) generally is not. Typically, they do not yield a closed-form solution and we must content ourselves with approximate solutions. There are essentially two different ways to tackle this problem: the *differentiate-then-discretize* approach, and the *discretize-then-differentiate* approach [2].

In a *differentiate-then-discretize* approach, one directly approximates the forward and backward equations using numerical schemes. Here, the approximation error to the gradient comes from the discretization error made in the solver for both the forward and backward equations. This method is used in the black box solvers presented in (Chen et al. 2018). It has the advantage of allowing the use of non-differentiable steps in the solver. However, this method can yield inconsistent gradients of cost functional $\mathcal{J}$, the discretization of the adjoint equations depends highly on the studied problem and therefore must be carefully selected and tuned (Carrassi et al. 2018).

In a *discretize-then-differentiate* approach, a differentiable solver for the forward equations is used, *e.g.* using an explicit Euler scheme $X^\theta_{t+\delta t} \approx X^\theta_t + \delta t F_\theta(X^\theta_t)$. Based on the solver's sequence of operations for the forward equations, the backward equations and the gradient can be directly obtained using automatic differentiation software (Paszke et al. 2017). This algorithm is actually equivalent

2. The differentiate-then-discretize method is often referred to as the *continuous adjoint method*, and the *discretize-then-differentiate* approach as the *discrete adjoint method* (Sirkes et al. 1997).

to backpropagation (LeCun et al. 1988) which can be derived as a special case of it. As the step-size approaches zero, the forward and backward equations are recovered.

It is important to note that, while the two methods are consistent and both converge to the equations derived in Theorem 4.1, they do not always yield the same results as the two approaches proceed differently. In our experiment, the second one proved more stable and the fact that we were limited to differentiable solvers experimentally was not an obstacle. This might not always be the case so the choice must be made after some exploration.

4.2.2 Models

We propose two variants of our models:

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.

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[3] Thus, in this setting, the dataset used here is of the form $\{X_0^{(i)}, Y_1^{(i)}, ..., Y_T^{(i)})\}$ 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.

4.3 Experiments

In this section we evaluate our model, both quantitatively and qualitatively. We consider two different datasets corresponding to two dynamical systems. We evaluate our method with respect to its ability to predict observations and to reproduce the dynamics of the hidden state.

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

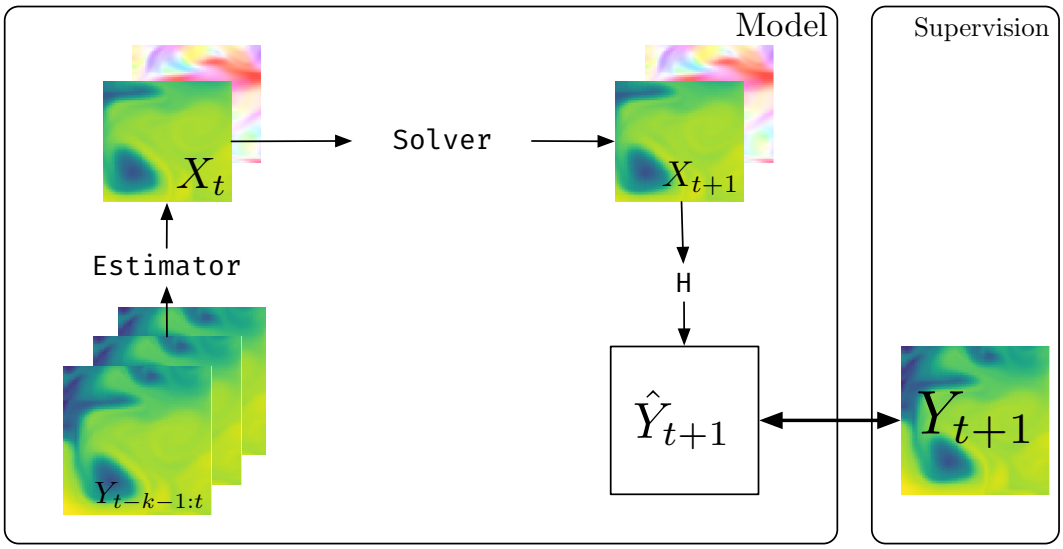


Figure 4.1 – Setting 1 : initial state is estimated from the input observation $Y_{t-k-1:t}$ and future states are estimated through the forecasting module. The partial observation Y is computed through the operator $\mathcal{H}$. The error signal is calculated as the Euclidean distance between $\hat{Y}_{t+1}$ and the real Y_{t+1}, and is backprogated through the operator scheme to correct the forecast and estimation module. To produce multiple time-step forecasts, the predicted observation is fed back in the model in an autoregressive manner.

4.3.1 Datasets.

Our two datasets are the following:

- **The Shallow Water equations** are derived from the Navier Stokes equations when integrating over the depth of the fluid (see supplementary material, Section 7.1.0.1). These equations are discretized on a spatial 80×80 grid. We decompose the simulation into train-validation and test subsets of 600 and 1000 acquisitions images respectively.

- **The Navier-Stokes equations** (see Section 7.1.0.2) are discretized on a spatial 64×64 grid. We use 15000 observations images for the train set and 10000 for the test.

4.3.2 Experimental setting.

In practice, the cost functional $\mathcal{J}$ is estimated on a minibatch of sequences from the dataset and optimized using stochastic gradient descent. For both datasets, the split between train, validation and test sets is made so that each

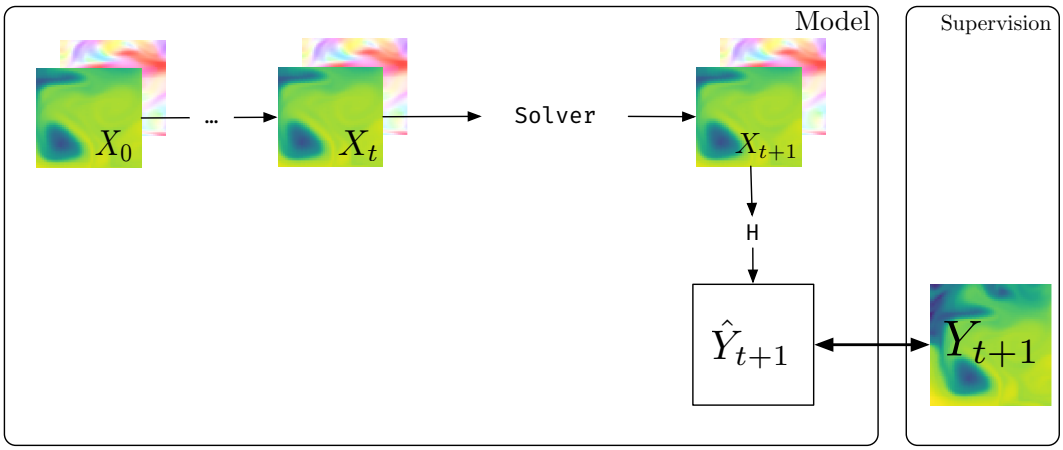


Figure 4.2 – Setting 2 : future state are estimated from the input state (X_0). The partial observation is computed through the operator $\mathcal{H}$. The error signal is calculated as the Euclidean distance between $\hat{Y}_{t+1}$ and the real Y_{t+1},, and is backprogated through the operator scheme to correct the forecast module. To produce multiple time-step forecasts, one simply apply the forecast F module on the estimated state.

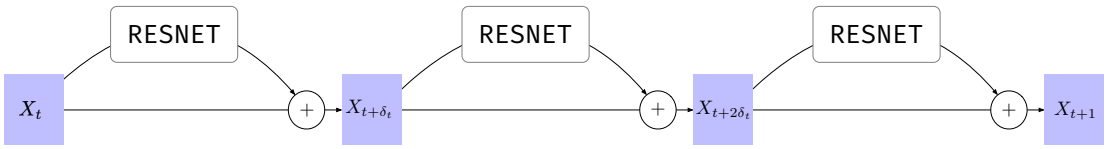


Figure 4.3 – The discretization of our forward equation : a three steps Euler scheme.

split only includes sequences generated by *different, independently sampled* initial conditions. The two datasets are completely simulated: we then have access to the true full state to initialize our algorithm X_0 in equation 4.5.

4.3.2.1 Implementations

Throughout *all* the experiments, F_θ is a standard convolutional residual network (He et al. 2015),, with 2 downsampling layers, 6 residual blocks, and bilinear up-convolutions instead of transposed convolutions. To discretize the forward equation 4.2 in time, we use a simple three steps Euler scheme (see Figure 4.3). For the spatial discretization, we use the standard gridlike discretization induced by the dataset. The weights of the residual network θ are initialized using an orthogonal initialization. Our model is trained using an 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).

4.3.2.2 Metrics.

To evaluate our model's performance we consider the quality of the predictions, using the renormalized mean-squared error between generated and ground-truth observations, averaged over the time sequence, and the spatial coordinates.

$$\frac{1}{K}\frac{1}{|\Omega|}\sum_{k=1}^{K}\sum_{x\in\Omega}\frac{\|\mathcal{H}(X_k(x))-Y_k(x)\|_2}{\|Y_k(x)\|_2} \tag{4.9}$$

To evaluate the quality of the hidden states, we use cosine similarity between the model's hidden state u and the truth hidden state of the system v [4]:

$$\frac{1}{K}\sum_{k=1}^{K}\frac{1}{|\Omega|}\sum_{x\in\Omega}\frac{\langle u(x),v(x)\rangle}{\|u(x)\|\,\|v(x)\|} \tag{4.10}$$

For the velocity vector field representation, color represents the angle, and the intensity the magnitude of the associated vectors (see Figure 3.6).

4.3.2.3 Models and Baselines

We use two different baselines:

- **PKN** This is the physics-informed deep learning model developed in Chapter 3, 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. The difference with the *Setting 1* model is that there is no forecast in the state space, but an explicit relation between the state X_t and Y_{t+1}. It does not model the full dynamical system but makes use of an autoregressive formulation.
- **PredRNN (Y. Wang et al. 2018)** This is a heavyweight, state-of-the-art model used for video prediction tasks. It is based on a Spatiotemporal Long-Short Term Memory (LSTM) that models spatial deformations and temporal variations simultaneously. As for PKN, it is an auto-regressive model.

4.3.3 Results.

4.3.3.1 Experiments with Navier Stokes equations

Forecasting Observations. Figure 4.4 shows a sample of the predictions of our system over the test set for the Navier Stokes equations. The good results it

4. We use the cosine similarity in order to compare with Prior Knowledge Network (PKN): the norm of its hidden state may not correspond to the ground truth norm.

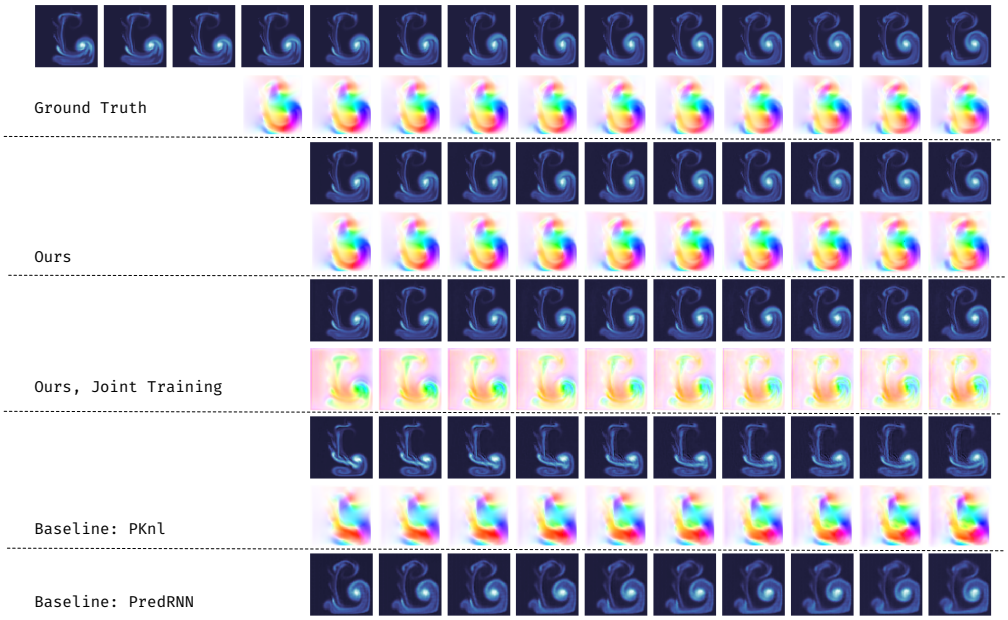


Figure 4.4 – Forecasting the Navier Stokes equations 10 time steps ahead with different models, starting from a given initial condition.

shows are confirmed by the quantitative results in Table 4.1. Our model is able to predict observations up to a long forecasting horizon (42 time steps), which means that it generalizes to horizon it has never seen (at is has been train on sequences of size 6), and thus has managed to learn the dynamical system. Note that the initial state used at test time has never been seen at training time which means that the optimization problem was solved correctly without overfitting. The cost function and supervision are only computed with observation, has described in our experiments setting. An interesting remark is to observe that the jointly trained model is slightly less accurate than the one that makes use of the initial state X_0.

Hidden State Discovery. Both our model forecasts a full state X_t and not only the observations Y_t. In order to predict the observations correctly, our model has to learn to predict future hidden states that contain information of the true state. By feeding the true initial conditions to our model, we find that our method is able to learn the true dynamics of the hidden state with a good accuracy, while never directly enforcing a penalty on the latter. Note that the only access our method has to full states is through the initial state provided as input. This result is intriguing: the model should theoretically be able to use a state *encoding* that is different from the one given by the initial condition. We hypothesize that our network's architecture is biased towards preservation of the input code.

Table 4.1 – Relative MSE and cosine similarity scores for our models and different baselines, at different temporal horizons on the Navier Stokes equations. As the PredRNN does not explicitly model the hidden state, we replace the cosine similarity scores for this baseline with XX.

Model	h=5		h=10		h=50	
	MSE	cosine	MSE	cosine	MSE	cosine
Setting 2	**0.118**	**0.798**	**0.180**	**0.679**	**0.628**	**0.483**
Setting 1	0.152	0.201	0.243	0.192	0.650	0.183
PKN	0.194	0.243	0.221	0.207	0.752	0.098
PredRNN (Y. Wang et al. 2018)	0.170	XX	0.227	XX	0.719	XX

Comparison with baselines. Visually, as can be seen in Figure 4.4 by looking at the small features of the observations, our model manages to capture many details which are important for robust long-term forecasts while the PredRNN model, which proves to be a strong baseline at the level of observations [5] for the first few steps, produces less sharp predictions which explains its worse performance when evaluated on long-term predictions. Other samples for long-term forecasts of our model can be seen in the appendix for the Navier Stokes as well as for the Shallow Water equations.

Figure 4.5 and Figure 4.6 show some examples of forecasts obtained with our model and confirm its accuracy over long-term predictions

In order to explore the properties of our model, we conduct an ablation study, whose results are reported in Section 4.3.3.3.

4.3.3.2 Experiments with Shallow Water equations

Table 4.2 shows the numerical evaluations for our model. Figures 4.7 and 4.8 show some examples of forecasts obtained with our model and confirm its accuracy over predictions, both for observations and velocities. Remember that the supervision is provided at the level of observations only.

Interpolation between data points. Our framework allows us to forecast for arbitrary times t, which means that if we only have access to samples at times $t, t+k, t+2k, \dots$ we can still predict observations at time $t, t+1, t+2, \dots$. This demonstrates the ability of our model to interpolate. Figure Figure 4.9 shows a sample of this interpolation mechanism. In this example, the model has

5. It does not produce meaningful hidden states.

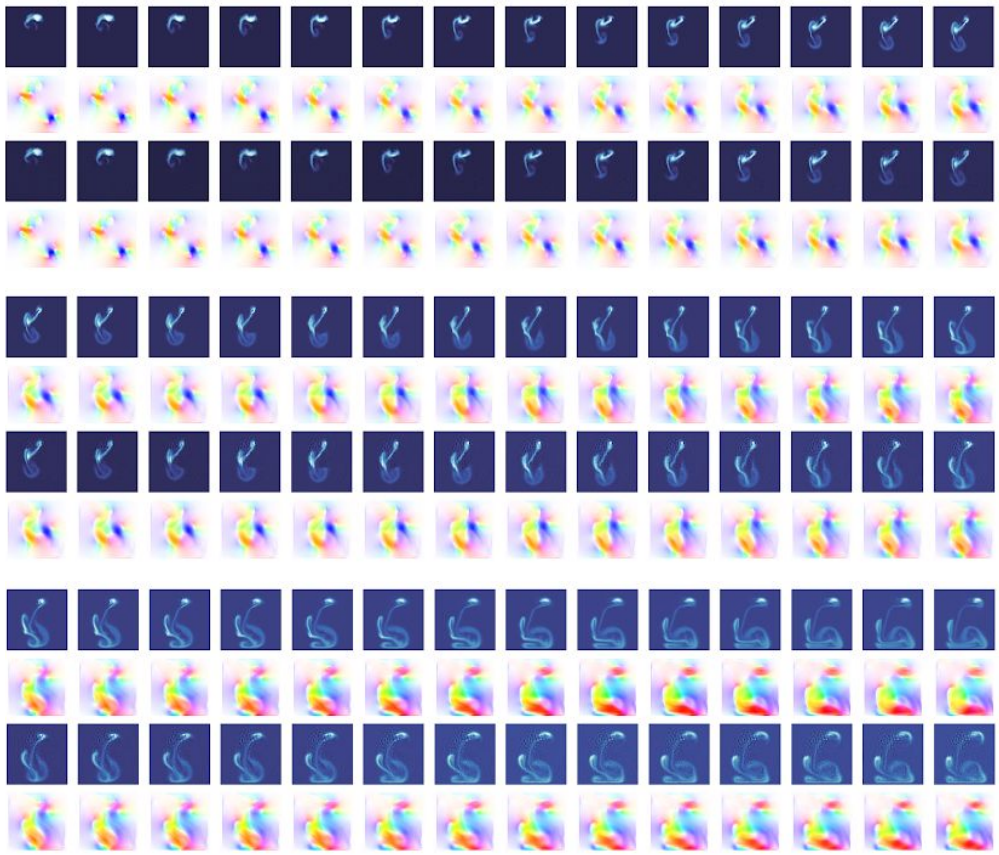


Figure 4.5 – Forecasting the Navier Stokes equations, starting from a given initial condition (not shown here). We forecast 42 time steps ahead. We show in this figure 3 different sequences of 42 time stted. Top 2 rows correspond to the ground truth and the bottom rows correspond model forecasts. Each sequences is represented as 4 consecutive rows.

been trained by regressing to the targets every 3 images (materialized on the figure by the red boxes). The outputs of the model are then compared with the unseen ground truth states. This shows that our approach allows us to learn the true evolution of the state. This is an important feature of our method, similar in this aspect to the claims of (Chen et al. 2018). It is applied here to a high-dimensional, highly non-linear and partially observed learned dynamical systems, for which we can interpolate the observations as well as the inferred hidden state.

4.3.3.3 Ablation Study

In order to better understand how our model works, we test different slightly modified versions of it on the Navier-Stokes dataset :

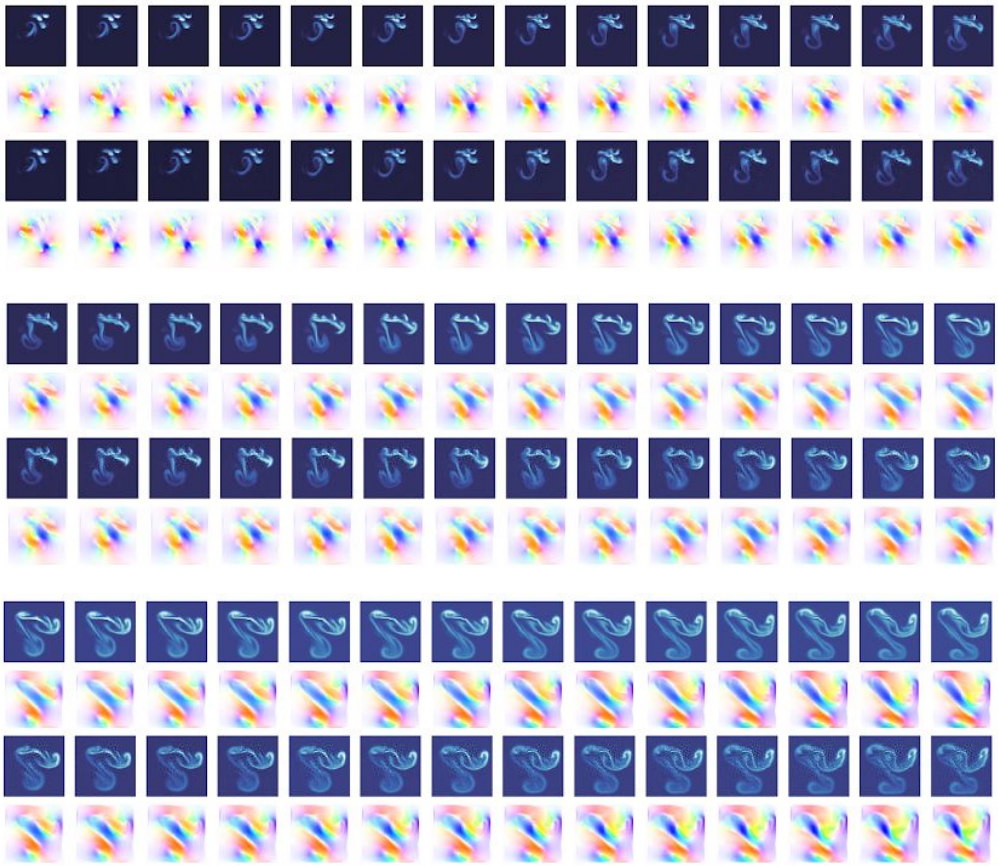


Figure 4.6 – Forecasting the Navier Stokes equations, starting from a given initial condition (not shown here). We forecast 42 time steps ahead. We show in this figure 3 different sequences of 42 time stted. Top 2 rows correspond to the ground truth and the bottom rows correspond model forecasts. Each sequences is represented as 4 consecutive rows.

Ours, Projection. Here we change the operator $\mathcal{H}$ and make it project to one dimension of the velocity field instead of on the pressure. We still give X_0 as input. The results, while slightly less good, are quite robust to this change, considering that we have not changed the hyperparameters of the model.

ResNet. Here we simply use a residual network, with *the exact same architecture* as the one used to parameterize our model. In other words, there are exactly the same number of parameters, layers,... as the F_θ which we learn and put into the solver. The results are notably less accurate for observations but, more importantly, this model turns out to be completely unable to forecast hidden states corresponding to the true ones. This shows that the way our

Table 4.2 – Relative MSE and cosine similarity scores for our model, at different temporal horizons on the Shallow Water equations

Model	h=5		h=10	
	MSE	cosine	MSE	cosine
Setting 2	0.1	0.995	0.12	0.992

model is structured around a solver which takes into account the differential structure of the studied problem is a strong regularizer.

ResNet no skip. This last argument may remind us that a residual network closely resembles the non-uniform discretization of an ODE. Thus, this should help it to perform well and explains the relatively good results on observations for the ResNet and, by getting rid of the skip connections while keeping all layers untouched, the performance should worsen. This is indeed what happens in our tests.

Unet. We tried using as F_θ this other classical architecture (Ronneberger et al. 2015), which is often used for regression problems, with roughly the same number of parameters as in our parameterization. It proved to be weak against our model.

Table 4.3 – Ablation study for our model, at different temporal horizons on the Navier Stokes equations

Model	h=5		h=10		h=50	
	MSE	cosine	MSE	cosine	MSE	cosine
Setting 2	**0.118**	**0.798**	**0.180**	**0.679**	0.628	0.483
Setting 1	0.191	0.432	0.288	0.620	**0.49**	**0.534**
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

4.4 Conclusion

We have introduced a general data-driven framework to predict the evolution of space-time processes, when the system is highly complex and non-linear and

the state is not fully observed. Assuming the underlying system follows a time-dependent differential equation, we estimate the unknown evolution term with a neural network. This is in a natural way to model continuous-time systems. We propose a learning algorithm for this model. Experiments performed on two simulated datasets from fluid dynamics show that the proposed method not only is able to produce high quality forecasts at different horizons, but also learns with a good accuracy the underlying state space dynamics.

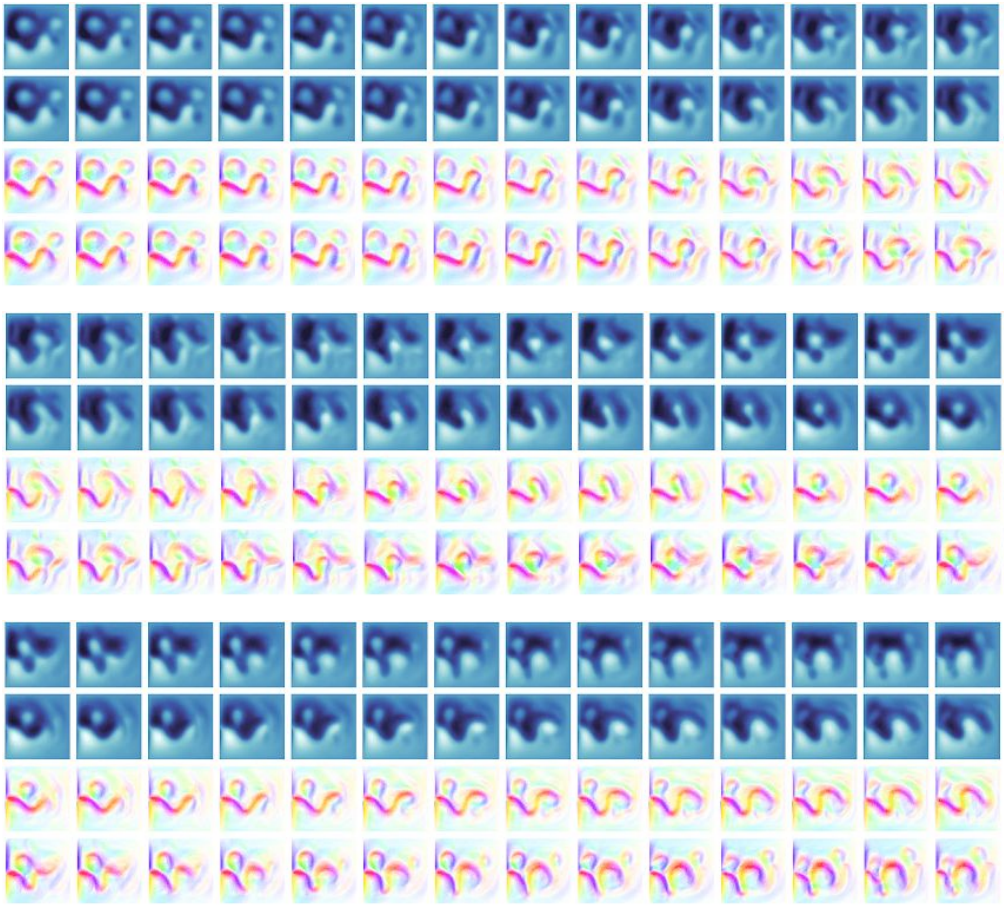


Figure 4.7 – Forecasting the shallow water equations, starting from a given initial condition (not shown here). We forecast 42 time steps ahead. We show in this figure 3 different sequences of 42 time steps. Top 2 rows correspond to the ground truth and the bottom rows correspond model forecasts. Each sequence is represented as 4 consecutive rows.

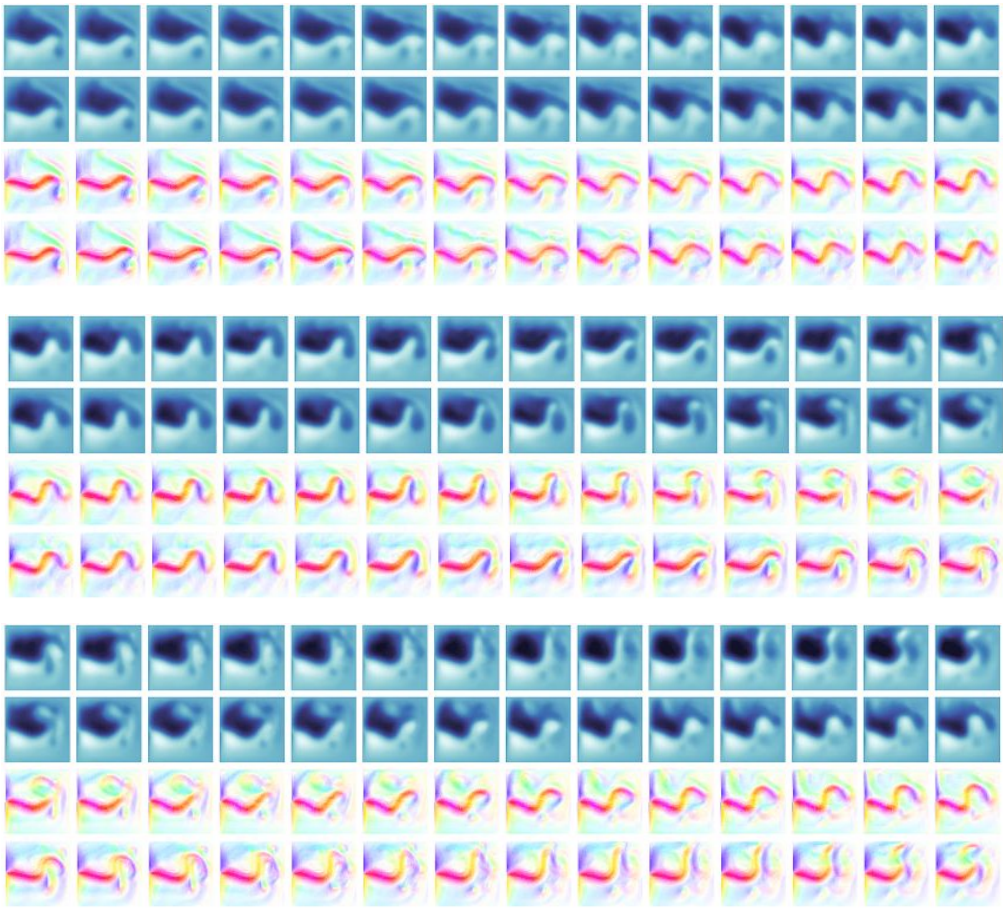


Figure 4.8 – Forecasting the shallow water equations, starting from a given initial condition (not shown here). We forecast 42 time steps ahead. We show in this figure 3 different sequences of 42 time steps. Top 2 rows correspond to the ground truth and the bottom rows correspond model forecasts. Each sequence is represented as 4 consecutive rows.

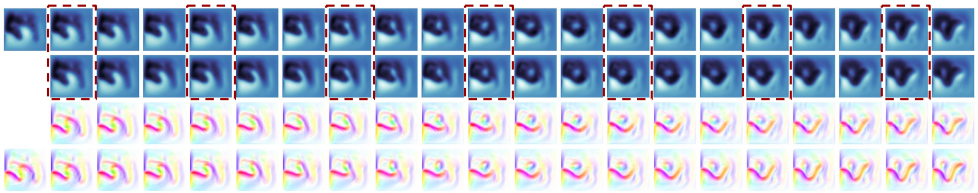


Figure 4.9 – Time interpolations with our approach on the test set. We train our model by regressing to the targets every 3 images (materialized by the red boxes). We then compare the outputs of the model with the unseen ground truth states.

CHAPTER 5

UNSUPERVISED IMAGE RECONSTRUCTION

Contents

5.1 Introduction **66**
5.2 Models **68**
5.2.1 Problem setting. 68
5.2.2 Approach 69
5.3 Method **70**
5.3.1 Handling the Likelihood Term 70
5.3.2 Handling the Prior Term 72
5.3.3 Putting everything together 73
5.4 Stochastic Variation **75**
5.4.1 Setting 76
5.4.2 Model 77
5.5 Temporal Variation **82**
5.5.1 Setting 82
5.5.2 Model 83
5.5.3 Prior handling 83
5.5.4 Likelihood Handling 84
5.6 Experiments **84**
5.6.1 Model architectures and Datasets 84
5.6.2 Baselines 90
5.6.3 Deterministic Results 94
5.6.4 Stochastic Results 95
5.6.5 Temporal Results 100
5.6.6 Comparison with Baselines 102
5.6.7 Ablation Study 104
5.7 Conclusion **106**
5.8 Additional Samples **107**

Chapter abstract

We address the problem of recovering an underlying signal from lossy, inaccurate observations in an unsupervised setting. Typically, we consider situations where there is little to no background knowledge on the structure of the underlying signal, no access to signal-measurement pairs, nor even unpaired signal-measurement data. The only available information is provided by the observations and the measurement process statistics. We cast the problem as finding the maximum a posteriori estimate of the signal given each measurement, and propose a general framework for the reconstruction problem. We use a formulation of generative adversarial networks, where the generator takes as input a corrupted observation in order to produce realistic reconstructions, and add a penalty term tying the reconstruction to the associated observation. We evaluate our reconstructions on several image datasets with different types of corruptions. The proposed approach yields better results than alternative baselines, and comparable performance with model variants trained with additional supervision. Finally, we present two extension of this work, by considering the special case of unsupervised inpainting. A first extension considers augmenting the observation space with a stochastic variable, in order to introduce some stochasticity in the reconstruction process. A second extension tackles the problem of inpainting occluded area in spatiotemporal sequences, such as cloud occluded satellite observations. The work in this chapter, in collaboration with Emmanuel De Bezenac and Yuan Yin, has led to the publication of a conference paper:

- Arthur Pajot, Emmanuel de Bezenac, and Patrick Gallinari (Sept. 2018). "Unsupervised Adversarial Image Reconstruction". In: URL: https://openreview.net/forum?id=BJg4Z3RqF7

- Yuan Yin, Arthur Pajot, Patrick Gallinari, and Emmanuel de Bézenac (Aug. 2019). "Unsupervised Inpainting for Occluded Sea Surface Temperature Sequences". In: *Climatinformatics Workshop*

5.1 Introduction

Many real world applications require acquiring information about the state of some physical system from incomplete and inaccurate measurements. For example, in infrared satellite imagery, one has to deal with the presence of clouds and a variety of other external factors perturbing the acquisition of temperature maps. This raises questions on how to recover the correct information and eliminate the contribution of external factors hindering the overall signal acquisition.

In this context, signal recovery does not usually yield a unique solution, meaning that multiple signal reconstructions could trivially explain the mea-

surements. For the above example, different missing temperature values could accurately explain the observations.

To cope with this indeterminacy, one usually relies on prior information on the structure of the true signal in order to constrain the reconstruction to plausible solutions (Stuart 2010). A common approach is to use handcrafted, analytically tractable priors, as in compressed sensing (Candès et al. 2005; Mota et al. 2017). This approach is limited to situations for which the underlying signal structure can be easily described, which are rarely observed in the wild.

Recent developments in generative models parameterized by neural networks (Goodfellow et al. 2014; Kingma et al. 2013; Dinh et al. 2016) offer a promising statistical approach to signal recovery, for which priors on the signal are not handcrafted anymore, but learned from large amounts of data. Despite exhibiting interesting results (Bora et al. 2017; Asim et al. 2018; Tripathi et al. 2018), these methods all require some form of supervision, either observation measurement-signal pairs, or at least unpaired samples from observations and underlying signals. For many practical problems, obtaining these samples is too expensive and/or impractical, which makes these approaches not suitable for such situations.

We address the problem of image and video reconstruction in an unsupervised setting, when only corrupted observations are available, together with some prior information on the nature of the measurement process. The learning problem is formulated as finding the maximum a posteriori estimate of signals given their measurements on the training set. We derive a natural objective for our reconstruction network, composed of a linear combination of an adversarial loss for recovering realistic signals, and a reconstruction loss to tie the reconstruction to its associated observation (Section 5.2.2). This model is evaluated and compared to baselines on 3 image datasets, CelebA (Z. Liu et al. 2015), LSUN Bedrooms (F. Yu et al. 2015), Recipe-1M (Marin et al. 2018), where we experiment with different types of measurement processes corrupting the images.

We also consider the special case of inpainting. While still a difficult task, unsupervised inpainting allows us to introduce an operator able to recover the mask of a given measurement. We then propose a first extension of our model where the observations are augmented by a stochastic variable, in order to learn the distribution of plausible reconstruction. We present a second extension where we consider inpainting video. This is up to our knowledge the first attempt to solve the problem of unsupervised video completion using general ML methods. This method is fully data driven and does not use any hand-defined analytical prior on the spatiotemporal sequence.

Our contributions are:

- A novel, computationally efficient framework for dealing with large scale signal recovery in an unsupervised context, applicable to a wide range of situations,
- A model and a new way of training a deep learning architecture for implementing this framework,
- An extension of this model in a stochastic setting, for unsupervised inpaiting.
- An extension where we inpaint sequences of images.
- Extensive evaluations on a number of image datasets with different measurement processes.

5.2 Models

Notations. We use capital letters (*e.g.* X) for random variables, and lower-case letters (*e.g.* x) for their values. $\mathrm{p}_X(x)$ denotes the distribution (or its density in the appropriate context) of X evaluated at x.

5.2.1 Problem setting.

Suppose there exists a signal $X \sim \mathrm{p}_X$ we wish to acquire, but we only have access to this signal through lossy, inaccurate measurements $Y \sim \mathrm{p}_Y$. The measurement process is modeled through a stochastic operator F mapping signals X to their associated observations Y. We will refer to F as the *measurement process*, which *corrupts* the input signal. F is parameterized by a random variable $\Theta \sim \mathrm{p}_\Theta$ following an underlying distribution p_Θ we can sample from, which represents the factors of corruption. Thus, given a specific signal x, we can simulate its measurement by first sampling θ from p_Θ, and then computing $F(x;\theta)$. Additional sources of uncertainty, e.g., due to unknown factors, can be modeled using additive *i.i.d.* Gaussian noise $\mathcal{E} \sim \mathcal{N}(0, \sigma^2 I)$, so that the overall observation model becomes:

$$Y = F(X;\Theta) + \mathcal{E} \tag{5.1}$$

F is assumed to be differentiable *w.r.t.* its first argument X, and Θ and X to be independent (denoted $X \perp\!\!\!\perp \Theta$). Different instances of F will be considered (refer to Section 5.6.1.6), like random occlusions, information acquisition from a sparse subset of the signal, overly smoothing out and corrupting the original distribution with additive noise, etc. In such cases, the factors of corruption Θ

might respectively represent the position of the occlusion, the coordinates of the acquired information, or simply the values of the additive noise.

5.2.2 Approach

Given an observation y, our objective is to find a signal $\hat{x}$ as close as possible to the associated true signal x. From a probabilistic viewpoint, it is natural to formulate the problem as finding the Maximum A Posteriori (MAP) estimate, which consists in selecting the most probable signal x^* under the posterior distribution $\mathrm{p}_{X|Y}(\cdot|y)$:

$$x^* = \arg\max_x \log \mathrm{p}_{X|Y}(x|y) \tag{5.2}$$

or equivalently:

$$x^* = \arg\max_x \log \mathrm{p}_{Y|X}(y|x) + \log \mathrm{p}_X(x), \tag{5.3}$$

where $\mathrm{p}_{Y|X}(y|x)$, see as a function of x for a given y, is the likelihood of the signal x given observation y, and $\mathrm{p}_X(x)$ is the prior probability evaluated at x. Therefore, a good reconstruction must be likely to have generated the data, *i.e.* yield high likelihood, and look realistic, *i.e.* yield high probability under the prior.

In the general case, calculating the likelihood term $\mathrm{p}_{Y|X}(y|x)$ requires marginalizing over the noise parameters Θ and this does not yield an analytic form. As for the prior $\mathrm{p}_X(x)$, it is unknown, and we have no access to samples from X since we are in an unsupervised setting: there is then no direct way to estimate p_X either. In the general case considered here, with no assumption on the form of the distributions, solving Equation (5.3) is up to our knowledge an open problem.

In the following sections, we will introduce an approach to deal with the likelihood term (Section 5.3.1), and the unknown prior term (Section 5.3.2) in order to provide an approximate solution to Equation 5.3 (Section 5.3.3). For that, we will formulate the problem as learning a mapping $G : \mathcal{Y} \to \mathcal{X}$ that links each measurement y to its associated MAP estimate x^* on the training set. The associated objective is then:

$$G^* = \arg\max_G \mathbb{E}_{\mathrm{p}_Y}\left\{ \log \mathrm{p}_{Y|X}(y|G(y)) + \log \mathrm{p}_X(G(y)) \right\} \tag{5.4}$$

Which is obtained by plugging $G(y) = x$ into equation 5.3 and taking the expectation with respect to the distribution of observations p_Y.

5.3 Method

From Equation 5.4, we see that a valid reconstruction mapping G must yield high probability for the likelihood and the prior. This will guide the design of an appropriate objective during the following section, where the reconstruction mapping G will be implemented using a neural network.

5.3.1 Handling the Likelihood Term

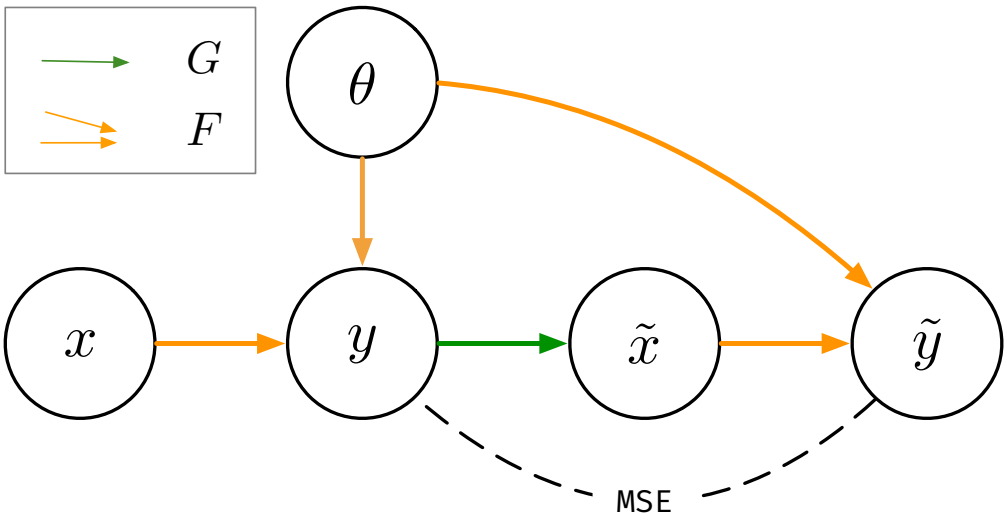


Figure 5.1 – The Figure illustrates the dependencies between the variables considered for handling the likelihood term when solving (5.4). The likelihood term in Equation 5.4 can be replaced by the expectation of $\frac{1}{2\sigma^2}\|y - F(G(y);\theta)\|_2^2$ (see Equation 5.10). To compute this expectation, one first simulates an observation y from a signal x using $F(x;\theta)$, then generates $\tilde{x} = G(y)$ and $\tilde{y} = F(\tilde{x};\theta)$, as in the Figure. This allows us to compute the MSE term in the above expression.

In the general case, evaluating the likelihood $\mathrm{p}_{Y|X}(y|x)$ in Equation 5.3 requires marginalizing on the unobserved noise variable Θ: $\mathrm{p}_{Y|X}(y|x) = \mathbb{E}_{\mathrm{p}_\Theta}\mathrm{p}_{Y|X,\Theta}(y|x,\theta)$, which involves computing an intractable integral. Most probabilistic model for images denoising make assumptions on the structure of the measurement operator $F(.,\Theta)$ and on the distribution of Θ in order to obtain an analytic form for the expectation (Demir et al. 2018; Boyat et al. 2015). Here, we consider more general measurement operators who do not necessarily lead to such a simplification and therefore proceed in a different way.

We outline below, the main steps of the method for handling the likelihood term $\mathrm{p}_{Y|X}(y|x)$ in Equation 5.4.

1. As X and Θ are independent, the expectation term $\mathbb{E}_{\mathrm{p}_Y} \log \mathrm{p}_{Y|X}(y|G(y))$ in Equation 5.4 can be rewritten as :

$$\mathbb{E}_{\mathrm{p}_\Theta \mathrm{p}_X \mathrm{p}_{Y|X,\Theta}} \log \mathrm{p}_{Y|X,\Theta}(y|G(y),\theta) + c_1 \tag{5.5}$$

, with c_1 constant w.r.t. G.

For all x, (in particular for $G(y)$), if X and Θ are independent, the log-likelihood can be decomposed as:

$$\begin{aligned} \log \mathrm{p}_{Y|X}(y|x) &= \log \mathrm{p}_{Y,\Theta|X}(y,\theta|x) - \log \mathrm{p}_{\Theta|X,Y}(\theta|x,y) \\ &\overset{X \perp\!\!\!\perp \Theta}{=} \log \mathrm{p}_{Y|X,\Theta}(y|x,\theta) + \log \mathrm{p}_\Theta(\theta) - \log \mathrm{p}_{\Theta|Y}(\theta|y) \end{aligned} \tag{5.6}$$

Applying the expectation *w.r.t* to the joint $\mathrm{p}_{Y,\Theta}$ on both sides, we obtain

$$\begin{aligned} \mathbb{E}_{\mathrm{p}_Y} \log \mathrm{p}_{Y|X}(y|x) &= \mathbb{E}_{\mathrm{p}_{Y,\Theta}} \{ \log \mathrm{p}_{Y|X,\Theta}(y|x,\theta) + \log \mathrm{p}_\Theta(\theta) - \log \mathrm{p}_{Y,\Theta}(\theta|y) \} \\ &= \mathbb{E}_{\mathrm{p}_{Y,\Theta}} \{ \log \mathrm{p}_{Y|X,\Theta}(y|x,\theta) \} + c_1 \end{aligned} \tag{5.7}$$

The terms $\log \mathrm{p}_\Theta(\theta)$ and $\log \mathrm{p}_{\Theta|Y}(\theta|y)$ do not depend on x, hence c_1 is a constant w.r.t. x. Plugging back $G(y)$ in place of x and applying the law of total probabilities w.r.t. X on the right-hand side, we obtain:

$$\mathbb{E}_{\mathrm{p}_Y} \log \mathrm{p}_{Y|X}(y|G(y)) = \mathbb{E}_{\mathrm{p}_\Theta \mathrm{p}_X \mathrm{p}_{Y|X,\Theta}} \log \mathrm{p}_{Y|X,\Theta}(y|G(y),\theta) + c_1 \tag{5.8}$$

2. The general measurement process described in Equation 5.1 induces $\log \mathrm{p}_{Y|X,\Theta}(y|G(y),\theta)$ to yield a simple analytic expression:

$$\log p(y|G(y),\theta) = -\frac{1}{2\sigma^2} \|y - F(G(y);\theta)\|_2^2 + c_2 \tag{5.9}$$

with c_2 constant.

3. The likelihood term $\mathbb{E}_{\mathrm{p}_Y} \log \mathrm{p}_{Y|X}(y|G(y))$ can then be replaced in the objective (5.4) by

$$-\mathbb{E}_{\mathrm{p}_\Theta \mathrm{p}_X \mathrm{p}_{Y|X,\Theta}} \frac{1}{2\sigma^2} \|y - F(G(y);\theta)\|_2^2 \tag{5.10}$$

Equation 5.10 shows that the likelihood term can be evaluated by first sampling a measurement y conditioned on a corruption parameter θ and signal x, and then constrain G such that $\|y - F(G(y); \theta)\|_2^2$ is close to zero.

Note that in this expression, the same parameter θ is used for simulating $\tilde{y}$ from $\tilde{x}$ and y from x (see Figure 5.1 and section 5.3.3 for more details). Unfortunately, this requires first sampling x from the signal distribution p_X which is unknown. In the following sections, we will see how we work around this problem.

5.3.2 Handling the Prior Term

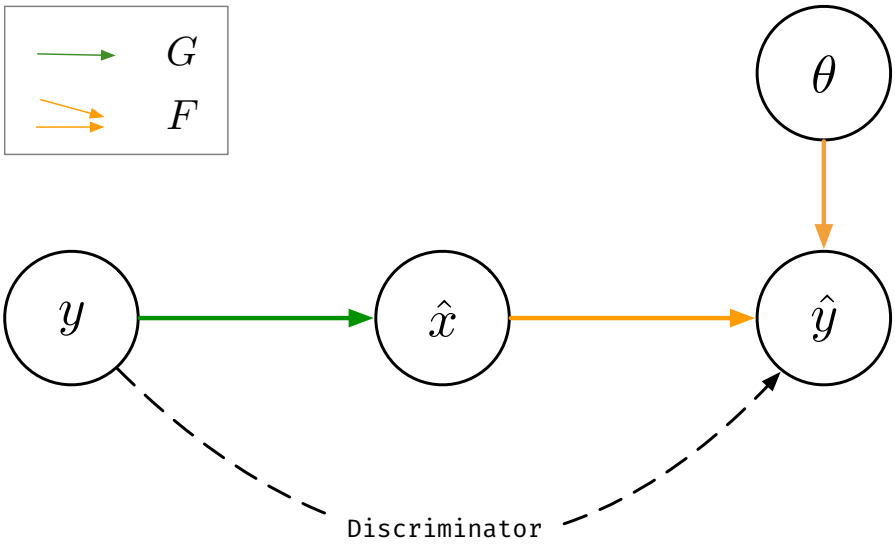


Figure 5.2 – The figure illustrates the dependencies of the variables used for dealing with the prior term in Equation 5.4. An observation y is sampled, and then transformed by the generative network into a reconstructed signal $\hat{x} = G(y)$. One then simulates a measurement $\hat{y} := F(\hat{x}; \theta)$ from this reconstruction. We then enforce the distributions of observations p_Y and simulated measurements p_Y^G to be similar using an adversarial loss. In order to produce indistinguishable distributions, the generator G has to *remove* the corruption and recover a sample $\hat{x}$ from p_X.

Maximizing w.r.t. the prior term $\mathrm{p}_X(G(y))$ in equation (5.4) is similar to learning a mapping G such that the distribution induced by $G(y)$, $\mathbb{E}_{\mathrm{p}_Y}\mathrm{p}_X(G(y))$ is close to the distribution p_X. The prior p_X being unknown, the only sources of information are the lossy measurements y and the known prior p_Θ on the

measurement process. In order to learn an approximation of the true prior p_X, we will use a form of generative adversarial learning, and build on an idea introduced in the AmbientGAN model by (Bora et al. 2018).

AmbientGAN aims at learning an *unconditional* generative model G of the true signal distribution p_X, when only lossy measurements y of the signal are available together with a known stochastic measurement operator F. In AmbientGAN, a generator is trained to produce uncorrupted signal samples from a latent code so that the generated signals when corrupted are indistinguishable from the observation measurements. In (Bora et al. 2018), the authors show that for some families of noise distributions p_Θ, the generator's induced distribution matches the signal's true distribution. Note that even if the generation process of the observations y in AmbientGAN is similar to the one considered in this paper (see Section 5.2.1), the objective is however different: when the aim of AmbientGAN is to learn a distribution of the underlying signal by sampling a latent space, ours is to reconstruct corrupted signals.

In order for G to produce uncorrupted signals, we will use an approach inspired from AmbientGAN, as illustrated in Figure 5.2. Given an observation y, one wants to reconstruct a latent signal approximation $\hat{x} = G(y)$ so that a corrupted version of this signal $\hat{y} = F(\hat{x})$ will have a distribution indistinguishable from the one of the observations y. The generator G and a discriminator D are trained on observations y and generated samples $\hat{y}$. The corresponding loss is the following [1]:

$$\mathcal{L}^{\text{prior}}(G) := \max_D \mathbb{E}_{Y\sim p_Y, \hat{Y}\sim p_Y^G}\{\log D(y) + \log\big(1 - D(\hat{y})\big)\} \tag{5.11}$$

where p_Y^G corresponds to the distribution induced by G's corrupted outputs ($\hat{y}$ in Figure 5.2) , *i.e.* $p_Y^G(y) := \mathbb{E}_{p_\Theta p_X^G}\{p(y|x,\theta)\}$ and p_X^G denotes the marginal distribution induced by G's outputs ($\hat{x}$ in Figure 5.2):

$$p_X^G(x) := \mathbb{E}_{p_Y} p_{X|Y}^G(x|y) = \mathbb{E}_{p_Y}\delta\big(x - G(y)\big)$$

This penalty enforces the marginal p_X^G to be close to the true prior distribution p_X, and thus forces G to map its input measurements onto p_X.

5.3.3 Putting everything together

In Section 5.3.1, we have shown that it is possible to maximize the average log-likelihood term in Equation 5.4, given that we can sample from the unknown prior distribution p_X. In Section 5.3.2, we have shown how it is possible to enforce the generator to produce samples from p_X, without ever having access to uncorrupted samples. The idea is then to use the distribution induced by the

1. the min term of the adversarial loss will be introduced later, see Equation (5.13)

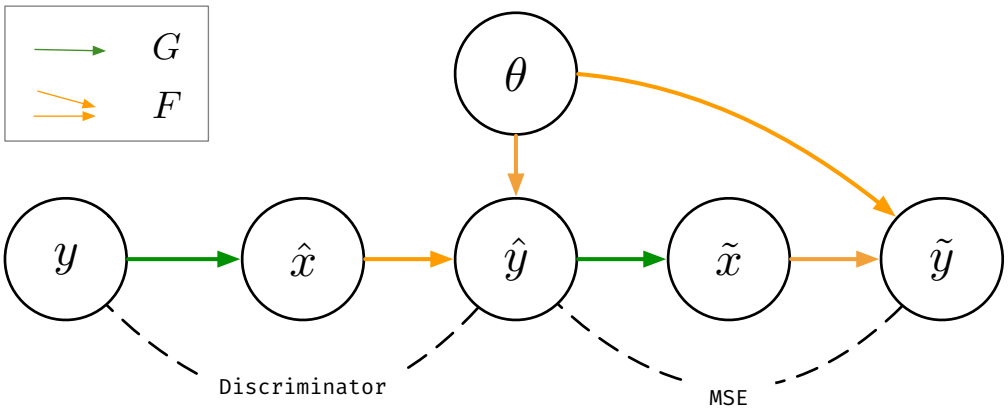


Figure 5.3 – General Approach. We wish to train G to recover a plausible signal from lossy measurements. As is shown in Section 5.2.2, this requires the reconstructions $\hat{x} := G(y)$ to have high probability under the likelihood and the prior. For simplicity, variable $\mathcal{E}$ has been omitted. *Prior:* we sample a measurement y from the data, produce a reconstruction $\hat{x}$, and sample a perturbation parameter θ. We enforce the simulated measurement $\hat{y} := F(\hat{x}; \theta)$ to be similar to measurements in the data using an adversarial penalty. Intuitively, this requires the network to *remove* the corruption. *Likelihood :* to enforce G to produce reconstructions with high likelihood, it is not possible to add a penalty to constrain the mean square error (MSE) between y and $\hat{y}$ to be small. This is because the underlying perturbation that caused y is unknown, and may be different from θ. Starting from $\hat{y}$ we generate a $\tilde{y}$ (see figure 5.3) using the same θ as the one used for generating $\hat{y}$. However, if $\hat{y}$ is similar to the measurements in the data, we can use it as a proxy for a measurement of the data where its associated θ is known. We then constrain $\|\hat{y} - \tilde{y}\|_2^2$ to be small ($\tilde{y} = F(G(\hat{y}))$.

generator's output p_X^G as a proxy for p_X to compute an approximate value of the expectation in Equation 5.5. This gives us the following penalty term:

$$\mathcal{L}^{\text{likeli}}(G) := \mathbb{E}_{\mathrm{p}_\Theta \mathrm{p}_X^G, \hat{Y} \sim \mathrm{p}_{Y|X,\Theta}} \|\hat{y} - F(G(\hat{y}); \theta)\|_2^2 . \tag{5.12}$$

The full objective is a linear combination of penalties (5.11) and (5.12):

We wish to learn a network that produces reconstructions that (1) are likely to have generated the data and (2) belong to the set of natural signals, by only accessing lossy, inaccurate measurements of the signal. To enforce (2), we sample a measurement y from the data, produce a reconstruction $\hat{x}$ such that its simulated measurement $\hat{y}$ must belong to the data distribution. Intuitively,

this requires the network to remove the corruption caused by the measurement process. This is not sufficient to enforce (1). Since the measurement y has been generated from a θ with the same underlying law p_Θ but not necessarily with the same valuation, we cannot directly minimize the distance between y and $\hat{y}$. However, if (2) is enforced, $\hat{y}$ is similar to the data, and we can use it as a substitute for a measurement of the data. We can then produce a reconstruction $\tilde{x}$ of the signal from $\hat{y}$, corrupt this signal again using the same θ that produced $\hat{y}$, and minimize the distance between $\hat{y}$ and its corrupted reconstruction $\tilde{y}$. For clarity, since the role of $\mathcal{E}$ is not essential to understanding the method from an abstract level, it has been omitted.

$$\arg\min_G \ \mathcal{L}^{\text{prior}}(G) + \lambda \cdot \mathcal{L}^{\text{likeli}}(G) \tag{5.13}$$

As illustrated by the dependencies highlighted in Figure 5.2, in the process of minimizing $\mathcal{L}^{\text{prior}}$, we sample from the marginal likelihood $\mathrm{p}_Y^G(y) := \mathbb{E}_{\mathrm{p}_\Theta \mathrm{p}_X^G}\{\mathrm{p}(y|x,\theta)\}$. The expectancy in the likelihood term $\mathcal{L}^{\text{likeli}}$ is precisely computed w.r.t. this distribution. We can then use the same samples in order to minimize the full objective (5.13). This gives us the Algorithm 5.1 described below, along with the dependency structure illustrated in Figure 5.3.

Algorithm 5.1 Training Procedure.

Initialize parameters of the generator G and the discriminator D.
while (G, D) not converged **do**
  Sample $\{y_i\}_{1\leq i\leq n}$ from data distribution p_Y
  Sample $\{\theta_i\}_{1\leq i\leq n}$ from P_Θ
  Sample $\{\varepsilon_i\}_{1\leq i\leq n}$ from $P_\mathcal{E}$
  Set $\hat{y}_i$ to $F(G(y_i), \theta_i) + \varepsilon_i$ for $1 \leq i \leq n$
  Update D by ascending:

$$\frac{1}{n}\sum_{i=1}^{n} \log D(y_i) + \log(1 - D(\hat{y}_i))$$

  Update G by descending:

$$\frac{1}{n}\sum_{i=1}^{n} \lambda \cdot \|\hat{y}_i - F(G(\hat{y}_i); \theta_i)\|_2^2 + \log(1 - D(\hat{y}_i))$$ [2]

end while

5.4 Stochastic Variation

In this section, we propose a variation of our model designed to handle the stochasticity of the data.

As mentioned above, image inpainting is usually an ill-posed problem so that multiple signal reconstructions could explain a corrupted image. Most approaches propose a unique image reconstruction among all the possible ones. A more challenging task consists in learning the distribution of the plausible reconstructions. A common approach to distribution learning consists in training a neural model to map a latent code taken from an easy-to-sample distribution, to a target output domain. By sampling the latent space, the image distribution can then be recovered. The generator performing the mapping is then supposed to learn to associate latent codes with some representations of the information missing in the observation.

For simplification's sake, we only consider the special case where the corruption is an inpainting corruption (Equation 5.14). This means that the likelihood handling is much simpler. However we handle the prior term in a similar fashion.

5.4.1 Setting

Like in Section 5.3, we suppose that there exists a domain of uncorrupted signals X with distribution p_X, but we only have access to incomplete measurements from the domain Y with distribution p_Y. We replace the *corruption process* by a *masking process* (we try to solve an inpainting probem) modeled through a stochastic operator $F : X \to Y$ mapping signals x to their associated observations y. We will refer to F as the measurement process. F is parameterized by mask $m \in M$ that we can sample with p_M, the mask distribution. Thus, given signal x, we can simulate the associated observation y by sampling m from p_M, and then calculating:

$$\boldsymbol{y} = F(\boldsymbol{x}, \boldsymbol{m}) = \boldsymbol{x} \odot \boldsymbol{m} + c \cdot \bar{\boldsymbol{m}} \tag{5.14}$$

where $m \sim \mathrm{p}_{\mathbf{M}}$ is an occlusion mask, generated from a known distribution with the same size as x and with components in $\{0, 1\}$, where 0 holds for a masked pixel. $\bar{m}$ denotes the complement of m, $\odot$ is the element-wise multiplication, all the masked pixels are supposed to be reset to a constant c which could be 0 or 1 depending on the observation process (see Section 5.6.1.6). Random variables $\mathbf{X}$ and $\mathbf{M}$ are assumed to be independent and F is assumed differentiable w.r.t. $\boldsymbol{x}$. In the following, we will suppose that one can retrieve the mask m directly from the observation y. This is not very restrictive since in most situations this is easy to do. We denote T the mask extractor $T(y) = m$.

Our goal is to learn the distribution of the semantically plausible reconstructions x for an observation y. Said otherwise, one wants a mapping G which, given an observation y and a random variable z produces a reconstruction of a plausible underlying signal x, i.e. $x = G(y, z)$. Since for a given y there are many

possible reconstructions x, one wants different samples z, to lead to different realizations x. The learned mapping G will associate to a given sample z, some information not contained in the observation y for generating a full image x. For example, if the generator is presented with a picture of a face with missing eyes, G could learn to associate to different z values, different representations of the color and shape of the eyes, thus generating diverse reconstructions x for the same observation y. We will adopt throughout this work a conditional adversarial approach, and G will then act as a generator. The model used for inference is illustrated in Figure 5.4.

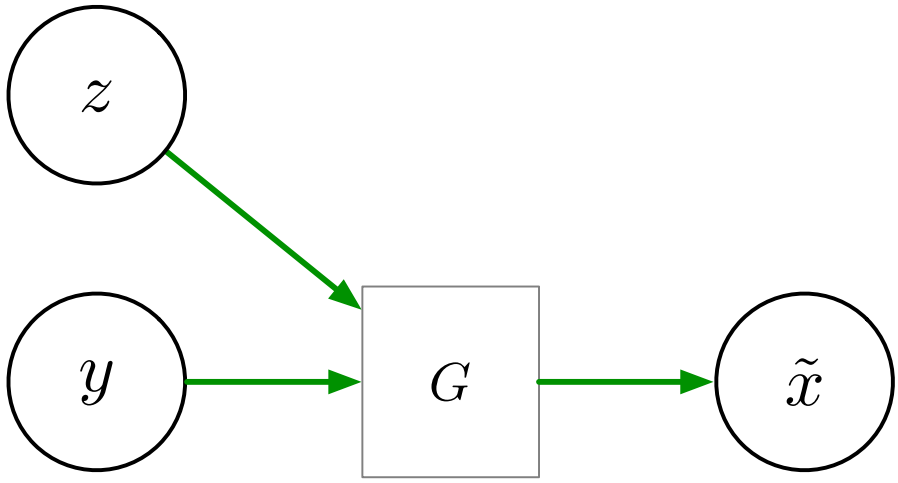


Figure 5.4 – A corrupted image y is sampled from Y together with a latent z from Z. The generator G then produces $\tilde{x} = G(y, z)$ a reconstructed image.

5.4.2 Model

5.4.2.1 Adversarial loss

Since we do not have access to the true images in X, but only to data sampled from partial observations in Y and to the measurement process F, a natural adversarial formulation of the problem is the following:

$$\mathcal{L}^{GAN}(G, D) := \mathbb{E}_{y \sim \mathrm{p}_Y}\left[\log D(y)\right] + \mathbb{E}_{\substack{y \sim \mathrm{p}_Y \\ m \sim \mathrm{p}_m \\ z \sim \mathrm{p}_z}}\left[\log\left(1 - D(F[G(y, z), m])\right)\right] \quad (5.15)$$

In this equation (see also Figure 5.5), $D(y)$ is a binary discriminator trained to separate observations from generated data, $G(y, z)$ is a reconstruction $\tilde{x}$ of the underlying signal x, $F[G(y, z), m]$ is the signal $\tilde{y}$ obtained from $\tilde{x}$ when applying mask m sampled from p_m. This mimics the measurement process: $\tilde{y}$ is obtained

from the generated $\tilde{x}$ in the same way as y is supposed to be obtained from the real x. Note that m being a stochastic variable, the m samples for y and $\tilde{y}$ will presumably be different. Given this loss function, training proceeds as usual by solving:

$$min_G max_D \mathcal{L}^{GAN}(G, D) \tag{5.16}$$

Equation 5.16 means that one wants to train a generator G so that the fake observation $\tilde{y} = F[G(y, z), m]$ has a distribution similar to the one of the observations y. The difference w.r.t. classical conditional GAN formulation is that since we are here in an unsupervised setting with only partial observations. Adversarial training is then performed on observations y and on their simulated reconstructions $\tilde{y}$ instead of on complete images x and $\tilde{x}$.

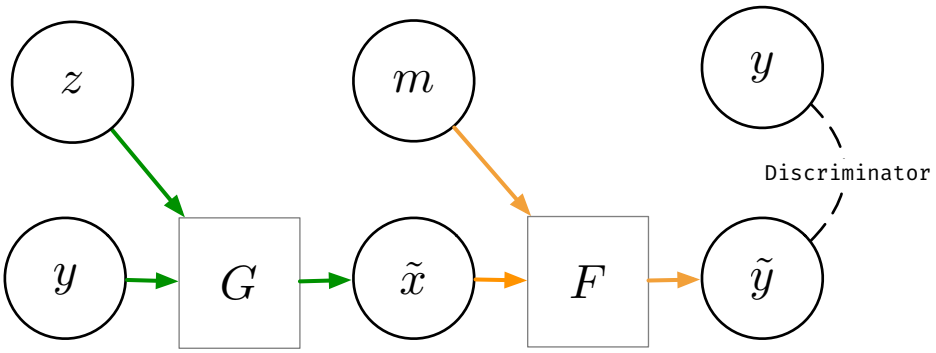


Figure 5.5 – Generative model for unsupervised learning of inpainting distributions. The discriminator is trained to differentiate between the generated $\tilde{y} = F(G(y, z), m)$ and samples y from the observation dataset Y.

This model holds for any stochastically generated noise or missing value. It could easily be refined for the inpainting task by focusing the image reconstruction process on the missing part of the image. In Equation 5.15, the whole image $\tilde{x}$ is supposed to be reconstructed by the generator G. Since one already knows y, there is no need to reconstruct it and one could simply plug y for x in the inpainting process. Let us suppose that one can retrieve the mask m from the observation y. This is not very restrictive since in most situations, this will amount at finding the pixels of y which equals to c . The reconstruction process then becomes:

$$\tilde{x} = G(y, z) \odot \bar{m} + y \tag{5.17}$$

This ensures that the observed part y remains unchanged in $\tilde{x}$. The loss in Equation 5.15 now becomes:

$$\mathcal{L}^{GAN}(G,D) := \mathbb{E}_{y\sim \mathrm{p}_Y}\big[\log D(y)\big] + \mathbb{E}_{\substack{y\sim \mathrm{p}_Y \\ m\sim \mathrm{p}_m \\ z\sim \mathrm{p}_z}}\big[\log\big(1 - D(F[G(y,z)+y, m])\big)\big]. \tag{5.18}$$

Figure 5.5 illustrates the corresponding adversarial model.

5.4.2.2 Latent Reconstruction Loss

Although the above model has the potential for learning inpainting distributions, in practice the generator trivially learns a deterministic mapping from observations y to reconstructed signals $\tilde{x}$. This has also been described in other settings, e.g., (Zhu et al. 2017b; Almahairi et al. 2018). We then propose two complementary losses which help to enforce the dependency on the stochastic component.

5.4.2.3 Encoding z loss.

A simple way to condition the generator G on the stochastic component z is to constrain the generator dependent outputs $\tilde{x}$ or $\tilde{y}$ to contain information from z. This could be implemented by training the model to recover z from the output $\tilde{y}$ for example. Let us denote $\tilde{z} := E(\tilde{y})$ this reconstruction with $E : Y \to Z$ a mapping to be learned. This simply amounts at adding the following term to the loss in Equation 5.18:

$$\mathcal{L}^{z}(G,E) := \mathbb{E}_{\substack{z\sim \mathrm{p}_z \\ y\sim \mathrm{p}_Y \\ m\sim \mathrm{p}_m}} \|z - E(F[G(y,z)+y), \bar{m}])\|_2^2 \tag{5.19}$$

This loss enforces the generator to use the information generated from the latent z into the reconstruction of the masked input. However, although this partially fulfills our goal, this is not sufficient, since as shown in (Almahairi et al. 2018) and (Chu et al. 2017), this auxiliary loss exhibits a "stenography" behavior: the corresponding model tends to hide information generated by G when using z as stochastic input, in a visually imperceptible way in the reconstructed image x. One needs a more direct way of enforcing diversity in the generated images. In order to alleviate this issue, we propose a second auxiliary loss.

5.4.2.4 Encoding y Loss.

An alternative to the above solution is to condition the z value to a sample y. Let us start from the model in Figure 5.5 and introduce a latent variable $\hat{z} = E(y)$ with E being as before a mapping from Y to Z. Let $\hat{z}$ and $\tilde{y}$, the latter being the output of the model in figure 5.5, be mapped successively to $\hat{x} = G(\tilde{y}, \hat{z})$ and $\hat{y} = F(\hat{x})$ as illustrated in figure 5.6. Let us then constrain $\hat{y}$ to be close to y via

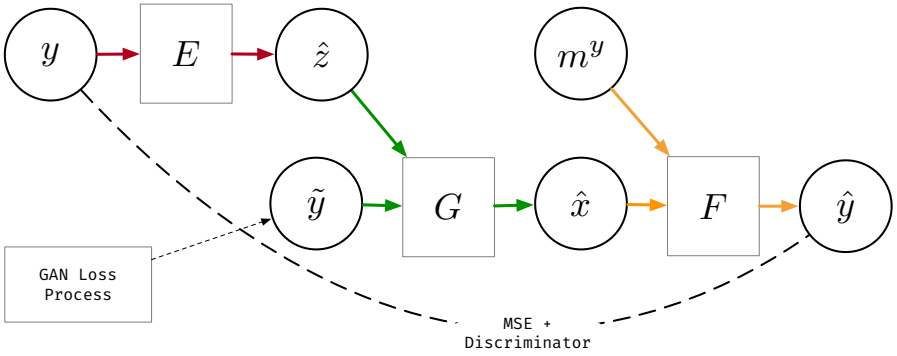


Figure 5.6 – Encoding y reconstruction loss. The input masked images y is encoded into the latent vector $\hat{z}$. The generator uses this code to generate a sample $\hat{x}$ which is masked with the mask m^y shall produce $\hat{y}$ close to y. The GAN loss process is the process generating $\tilde{y} \sim p_Y$ (as in Figure 5.5).

a Squared Error (MSE in Figure 5.6), and let their distribution be similar via an adversarial loss (Discriminator in Figure 5.6)). Then G will be forced to use both $\hat{z}$ and $\tilde{y}$. Let us briefly examine why. Thanks to the adversarial losses, $\hat{y}$ and y will have the same distribution. However they are different since they are associated with different samples m. In order to have $\hat{y}$ close to y, the generator G will then be forced to use $\hat{z}$. Using the above notations (see also Figure 5.6), the corresponding auxiliary loss is:

$$\begin{aligned}\mathcal{L}^y(G,D,E) = &\mathbb{E}_{y\sim p_Y}\left[\log D(y)\right] \\ &+ \mathbb{E}_{\substack{\tilde{y}\sim p_Y \\ \hat{z}\sim E(y)}}\left[\log\left(1 - D(F[G(\tilde{y},\hat{z}) + \tilde{y}), m^y]\right)\right] \\ &+ \mathbb{E}_{\substack{y\sim p_Y \\ \hat{z}\sim E(y)}}\left\|y - F[G(\tilde{y},\hat{z}) + \tilde{y}), m^y]\right\|_2^2\end{aligned} \tag{5.20}$$

In this equation, $\hat{z} \sim E(y)$ means that $\hat{z}$ is sampled from the distribution of Z generated by sampling $E(y)$. m^y denotes the mask extracted from y (see Section 5.4.2.1). For simplification, we have omitted in the second and third expectation terms, samplings needed to generate $\tilde{y}$.

This auxiliary loss is to be used in conjunction with loss in equation 5.18. The reconstructed image is, as before, $\tilde{x}$.

5.4.2.5 Putting It All Together

The two auxiliary losses (5.19) and (5.20) represent alternatives for enforcing the use of z. We have tested the two losses separately, each of them improves the quality and the diversity of the reconstructed images. We have found out

that combining them improved further the image quality and diversity over individual solutions (see Section 5.6.4). This is coherent with the finding of (Zhu et al. 2017b). The complete objective is then a combination of losses (5.18) (5.19) and (5.20):

$$\mathcal{L}(G, D, E) = \mathcal{L}^{GAN}(G, D) + \lambda_z \mathcal{L}^z(G, E) + \lambda_y \mathcal{L}^y(G, D, E) \tag{5.21}$$

Where the values of λ_z, λ_y are selected on a validation set. We describe the complete training process in the algorithm Algorithm 5.2.

Algorithm 5.2 Training Procedure.

Require: Initialize parameters of the generator G, the discriminator D and the encoder E.

while (G, D, E) not converged **do**

Sample $\{y_i\}_{1 \leq i \leq n}$ from data distribution p_Y

Sample $\{\tilde{m}_i\}_{1 \leq i \leq n}$ from p_m

Sample $\{z_i\}_{1 \leq i \leq n}$ from p_Z

Compute $m_i^y = T(y_i)$

Set $\tilde{x}_i \leftarrow G(y_i, z_i) \odot \bar{m}_i + y_i$ for $1 \leq i \leq n$

Set $\tilde{y}_i \leftarrow F(\tilde{x}_i, \tilde{m}_i)$ for $1 \leq i \leq n$

Set $\hat{z}_i \leftarrow E(\tilde{y}_i \odot \bar{m}^y)$ for $1 \leq i \leq n$

Set $\tilde{z}_i \leftarrow E(y_i)$ for $1 \leq i \leq n$

Set $\hat{x}_i \leftarrow G(\tilde{y}_i, \tilde{z}_i) \odot \tilde{m}_i + \tilde{y}_i$ for $1 \leq i \leq n$

Set $\hat{y}_i \leftarrow F(\hat{x}_i, m_i^y)$ for $1 \leq i \leq n$

Update D by ascending:

$$\frac{1}{n} \sum_{i=1}^{n} \log D(y_i) + \log(1 - D(\tilde{y}_i))$$

Update G and E by descending simultaneously:

$$\frac{1}{n} \sum_{i=1}^{n} \log(1 - D(\tilde{y}_i))$$

and

$$\frac{1}{n} \sum_{i=1}^{n} \lambda_{rec_y} \cdot \|\hat{y}_i - y_i\|_2^2 + \log(1 - D(\hat{y}_i))$$

and

$$\frac{1}{n} \sum_{i=1}^{n} \lambda_{rec_z} \cdot \|\hat{z}_i - z_i\|_2^2$$

end while

5.5 Temporal Variation

In this section, we also consider the case of unsupervised video reconstruction. We propose a model which can be used on different types of image sequences, physical or natural videos. We restrain ourselves to the special case of inpainting or video occlusion.

Our method does not make any assumption on the nature of the image sequence, it does not require any prior knowledge like most methods used for physical images do. This is up to our knowledge the first attempt to solve the problem of unsupervised video completion using general Machine Learning (ML) methods. This method is fully data driven and does not use any hand-defined analytical prior on the sequence of images.

5.5.1 Setting

We suppose that there exists an unknown spatiotemporal sequence $\boldsymbol{x} \sim \mathrm{p}_{\mathbf{X}}, \boldsymbol{x} \in \mathbb{R}^{C \times T \times H \times W}$, where $\boldsymbol{x}$ is a tensor denoting a C-channel sequence composed of T frames of $H \times W$ pixels. We denote $\boldsymbol{x}_t$ the t-th frame of the sequence and $\boldsymbol{x}_{t_1}^{t_2}$ the subsequence from the t_1-th to the t_2-th frame inclusive. With this notation, $\boldsymbol{x} = \boldsymbol{x}_1^T$. We do not have access to the original signal $\boldsymbol{x}$ but only to corrupted observation sequences of this signal $\boldsymbol{y} \sim \mathrm{p}_{\mathbf{Y}}, \boldsymbol{y} \in \mathbb{R}^{C \times T \times H \times W}$. Our objective is to reconstruct $\boldsymbol{x}$ from the corresponding observation $\boldsymbol{y}$. For example, $\boldsymbol{x}$ can be sea surface temperature (Sea Surface Temperature (SST)) at successive times while image sequence $\boldsymbol{y}$ is SST measurements via IR satellites occluded by moving clouds. We will suppose that $\boldsymbol{y}$ is obtained from $\boldsymbol{x}$ via a measurement process modeled through a stochastic operator F as follows:

$$\boldsymbol{y} = F(\boldsymbol{x}, \boldsymbol{m}) = \boldsymbol{x} \odot \boldsymbol{m} + c \cdot \bar{\boldsymbol{m}} \tag{5.22}$$

where $\boldsymbol{m} \sim \mathrm{p}_{\mathbf{M}}$ is an occlusion mask, generated from a known distribution with the same size as $\boldsymbol{x}$ and with components in $\{0, 1\}$, where 0 holds for a masked pixel. $\bar{\boldsymbol{m}}$ denotes the complement of $\boldsymbol{m}$, $\odot$ is the element-wise multiplication, all the masked pixels are supposed to be reset to a constant c which could be 0 or 1 depending on the observation process (see Section 5.6). Random variables $\mathbf{X}$ and $\mathbf{M}$ are assumed to be independent and F is assumed differentiable w.r.t. $\boldsymbol{x}$. In the following, we will suppose that one can retrieve the mask $\boldsymbol{m}$ directly from the observation $\boldsymbol{y}$. This is not very restrictive since in most situations this is easy to do. We denote T the mask extractor $T(\boldsymbol{y}) = \boldsymbol{m}$.

As in Section 5.2.1, our objective is then to recover the sequence $\boldsymbol{x}$ from the observations $\boldsymbol{y}$ and the corresponding binary masks $\boldsymbol{m}$. Adopting a probabilistic viewpoint, we want to select a reconstruction $\boldsymbol{x}^*$ which is the most plausible under the posterior distribution $\mathrm{p}_{\mathbf{X}|\mathbf{Y}}(\cdot|\boldsymbol{y})$.

5.5.2 Model

As in the Section 5.2.2, we formulate the problem as finding the most probable sequence conditioned on observations:

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}} \log \mathrm{p}_{\mathbf{X}|\mathbf{Y}}(\boldsymbol{x}|\boldsymbol{y}) = \arg\max_{\boldsymbol{x}} \log \mathrm{p}_{\mathbf{X}}(\boldsymbol{x}) + \log \mathrm{p}_{\mathbf{Y}|\mathbf{X}}(\boldsymbol{y}|\boldsymbol{x}) \tag{5.23}$$

To tackle these issues, we then again introduce a mapping $G : \mathbf{Y} \mapsto \mathbf{X}$, parameterized by a neural network φ and associating measurement $\boldsymbol{y}$ to its estimate $\boldsymbol{x}$. G will allow us to approximate the underlying distribution of training sequences.

By plugging $G(\boldsymbol{y})$ into Equation 5.23, the objective becomes:

$$G^* = \arg\max_{G} \underbrace{\mathbb{E}_{\boldsymbol{y}\sim \mathrm{p}_{\mathbf{Y}}}[\log \mathrm{p}_{\mathbf{X}}(G(\boldsymbol{y}))]}_{\text{prior}} + \underbrace{\mathbb{E}_{\boldsymbol{y}\sim \mathrm{p}_{\mathbf{Y}}}[\log \mathrm{p}_{\mathbf{Y}|\mathbf{X}}(\boldsymbol{y}|G(\boldsymbol{y}))]}_{\text{likelihood}} \tag{5.24}$$

5.5.3 Prior handling

Let us first handle the prior term in Equation 5.24. We want the distribution induced from $G(\boldsymbol{y})$ to be close to $\mathrm{p}_{\mathbf{X}}$. In order to do so, we will use an adversarial approach. We will build on the ideas introduced in Bora et al. 2018; Pajot et al. 2018 for still images. The process is illustrated in Figure 5.3. For a given observation $\boldsymbol{y}$, we want to generate an approximation of the unknown true sequence $\hat{\boldsymbol{x}} \equiv G(\boldsymbol{y})$. The prior $\mathrm{p}_{\mathbf{X}}$ being unknown, the only available information source is the observation $\boldsymbol{y}$ and the noise prior $\mathrm{p}_{\mathbf{M}}$. For a given generated signal $\hat{\boldsymbol{x}}$, we compute a corrupted version of $\hat{\boldsymbol{x}}$ through the known mask $\hat{\boldsymbol{m}}$, $\hat{\boldsymbol{y}} \equiv F(\hat{\boldsymbol{x}}, \hat{\boldsymbol{m}})$ with $\hat{\boldsymbol{m}} \sim \mathrm{p}_{\mathbf{M}}$. We will train G to make the distributions of $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ indistinguishable. In order to succeed, the generator G will have to remove the corruption from $\boldsymbol{y}$ and recover a sample $\hat{\boldsymbol{x}}$ from the distribution $\mathrm{p}_{\mathbf{X}}$. Generator G will then act as an inpainter conditioned on $\boldsymbol{y}$. This will enforce the distribution of the reconstructed sequences $\hat{\boldsymbol{x}}$ to be close to the distribution of true ones $\boldsymbol{x}$ and maximize the prior term.

A direct application of the adversarial training idea suggests using a discriminator operating directly on the sequences. We found out that using an additional discriminator on frames worked better than using a unique one operating on sequences. We then use two discriminators D_s and D_f respectively associated with whole sequences and with individual frames to optimize G. D_s

separates sequences $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$. D_f distinguishes real frames $\boldsymbol{y}_t$ from fake ones $\hat{\boldsymbol{y}}_t$. The loss function used for training G, D_s, and D_f is:

$$\min_G \mathcal{L}(G) = \max_{D_s, D_f} \mathbb{E}_{\boldsymbol{y}\sim \mathrm{p}_{\mathbf{Y}}, \hat{\boldsymbol{y}}\sim \mathrm{p}_{\mathbf{Y}}^G}[\log D_s(\boldsymbol{y}) + \log(1 - D_s(\hat{\boldsymbol{y}})) + \frac{1}{T}\sum_{t=1}^{T} \log D_f(\boldsymbol{y}_t) + \log(1 - D_f(\hat{\boldsymbol{y}}_t))] \tag{5.25}$$

with $\mathrm{p}_{\mathbf{Y}}^G(\boldsymbol{y}) \equiv \mathbb{E}_{\boldsymbol{m}\sim \mathrm{p}_{\mathbf{M}}, \boldsymbol{x}\sim \mathrm{p}_{\mathbf{X}}^G}[\mathrm{p}_{\mathbf{Y}|\mathbf{X},\mathbf{M}}(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{m})]$, corresponding to the distribution of the corrupted sequences $\hat{\boldsymbol{y}}$ generated via the measurement operator F. $\mathrm{p}_{\mathbf{X}}^G(\boldsymbol{x})$ is the distribution of $\hat{\boldsymbol{x}}$ induced by G from $\boldsymbol{y}$, i.e. $\hat{\boldsymbol{x}} = G(\boldsymbol{y})$.

5.5.4 Likelihood Handling

Let us now handle the likelihood term in Equation 5.24:

$$\mathbb{E}_{\boldsymbol{y}\sim \mathrm{p}_{\mathbf{Y}}}[\log \mathrm{p}_{\mathbf{Y}|\mathbf{X}}(\boldsymbol{y}|G(\boldsymbol{y}))]. \tag{5.26}$$

This likelihood is maximized when we are able to perfectly reconstruct $\boldsymbol{y}$ from $G(\boldsymbol{y})$. One way to ensure this property is to constrain G to directly use $\boldsymbol{y}$ for the non-occluded area of the reconstructed image $G(\boldsymbol{y})$. This can be easily achieved through the following mapping:

$$G(\boldsymbol{y}) \equiv \varphi(\boldsymbol{y}) \odot \bar{\boldsymbol{m}} + \boldsymbol{y} \odot \boldsymbol{m} \tag{5.27}$$

where φ is an NN responsible for reconstructing the missing part of $\boldsymbol{y}$, $\boldsymbol{m} = T(\boldsymbol{y})$ is the mask retrieved from $\boldsymbol{y}$. G maps $\mathbf{Y}$ to $\mathbf{X}$ with the help of mask $\boldsymbol{m}$ to ensure that the network will only generate values for occluded pixel, while keeping all the information from $\boldsymbol{y}$. To summarize, optimizing the prior term amounts at training φ for inputting the missing pixels while optimizing the likelihood term is simply achieved by copying the non-occluded portion of $\boldsymbol{y}$.

5.6 Experiments

5.6.1 Model architectures and Datasets

Model selection for unsupervised learning is an open problem. For all three settings (image deterministic, image stochastic and video), we suppose that a few full observed samples are available for model selection. We thus selected our models by evaluating our model on a small subset of our data (our validation set). The results on Section 5.6.3 are reported on a test set.

5.6.1.1 Deterministic Image Model

For our reconstruction network G, we use the image to image ResNet architecture used in (Isola et al. 2016; Zhu et al. 2017b). For the discriminator D, we used the Patch-GAN architecture (Zhu et al. 2017b; Isola et al. 2016). For the generator, we have also found that using an image-to-image variant of the recent Self-Attention GAN architecture (Zhang et al. 2018) stabilized training and improved overall performance on a number of measurement processes.

As in (Zhang et al. 2018), we use imbalanced learning rates for the generator and the discriminator (0.0001 and 0.0004, respectively), using the Adam optimizer ((Kingma et al. 2014)), using $\beta_1 = 0$ and $\beta_2 = 0.9$. The weights are initialized using orthogonal initialization. We set $\lambda = 2$, and exponentially decay the learning rate every 400 iteration, setting the decay factor to 0.995.

5.6.1.2 Stochastic Image Model

All models are trained using Adam (Kingma et al. 2014) with $\beta_1 = 0$ and $\beta_2 = 0.99$. The batch size is set to 128, for all experiments. We only update the networks every 4 step, artificially setting the batch size to 512. In order to train the generative networks, we use the non-saturating adversarial hinge loss (Miyato et al. 2018) (this loss is described in Chapter 2).

The networks used are similar to the ones described in Section 5.6.1.1.

5.6.1.3 Temporal Model

We briefly describe our model architecture and training setting. Our model utilizes a ResNet-type self-attention network (Zhang et al. 2018) for the generator G, composed of 3D-ResNet blocks and spatial self-attention layers. Spatial discriminator D_s is a 2D convolutional NN for binary classification. Temporal discriminator D_t uses the same structure as D_s but with 3D convolutions.

We apply hinge loss for as in (Zhang et al. 2018). All three networks are trained using Adam optimizer with a learning rate of 1×10^{-4} and $(\beta_1, \beta_2) = (0, 0.999)$ with batch size 1. All networks are initialized with normal distributions with a gain of 0.02 and have an architecture similar to the one described in Section 5.6.1.1.

5.6.1.4 Still Image Dataset

We evaluate our approach using three different image datasets :

CelebA. (Z. Liu et al. 2015). Dataset of celebrities, containing approximately 200 000 samples. As (Bora et al. 2018), the images are center-cropped.

LSUN Bedrooms (F. Yu et al. 2015). Dataset of bedrooms, containing 3 million samples.

Recipe-1M (Marin et al. 2018). Dataset of cooked meals, containing approximately 600 000 samples.

All the images have been resized to 64×64 for the standard experiment and 128×128 for the stochastic experiment. In order to place ourselves in the most realistic setting possible, every image has been corrupted once, *i.e.* there is never multiple occurrences of an image corrupted with different corruption parameters.

5.6.1.5 Video Dataset

SST The Sea Surface Temperature dataset used for the experiments includes 2 subsets of GLOBAL Sea Physical Analysis and Forecasting Product [3] from E.U. Copernicus Marine Service Information. This is a monitor system providing simulated but realistic global ocean SST data, which integrates satellite-derived and *in situ* data by assimilation. Our dataset is a part of the hourly mean SST, the finest timescale we have access to. The data we use is a part of the archive of analysis integrating real-world data. We retrieved our training-and-validation set and test set respectively from two different marine regions.

FaceForensics++ (Rössler et al. 2019) This dataset contains 1000 videos of non-occluded face movements on a static background. It was initially created for forgery detection. In our case, we extracted the faces from the original unforged videos with `face_recognition` [4], thus keeping only the changing component of the videos. The faces have been cropped and resized to 64×64.

KTH (Schuldt et al. 2004) A human action dataset containing 2391 video clips of 6 human actions [5]. The videos have been recorded with 25 subjects in different environments. All frames have been resized to 64×64.

BAIR Robot Pushing Dataset (Ebert et al. 2017) This dataset contains 44374 videos recorded by an one-armed robot [6]. It pushes objects and changes movement direction in a stochastic manner. All videos share similar tabletop with static background. All frames have been resized to 64×64.

3. http://marine.copernicus.eu/services-portfolio/access-to-products/
4. https://github.com/ageitgey/face_recognition
5. http://www.nada.kth.se/cvap/actions/
6. https://sites.google.com/view/sna-visual-mpc

5.6.1.6 Corruptions

Let us present the different measurement processes F used in the experiments, also named corruptions:

Remove-Pixel. This measurement process randomly samples a fraction p of pixels uniformly and sets the associated channel values to 0. All the corresponding channel values are set to 0.

Remove-Pixel-Channel. Instead of setting to 0 a pixel for all channels as in Remove-Pixel, one samples a pixel coordinate and a channel, and sets the corresponding value to 0.

Convolve-Noise. Here $F(x;\theta) := k * x + \theta$, where $*$ is the convolution operator and k is a mean filter of size l. For each pixel, noise θ sampled from a zero-mean Gaussian of variance σ_C^2 is added to the previous result.

Patch-Band. A horizontal band of height h whose vertical position in the image is uniformly sampled from the set of possible positions. For each pixel falling inside the band, its associated value is set to 0. The resulting measurement for pixel at column i and row j can be summarized as:

$$F(x;\theta)_{i,j} := \begin{cases} 0, & \text{if } j \in \{\theta, \dots, \theta + h\} \\ x_{i,j}, & \text{otherwise} \end{cases} \tag{5.28}$$

where θ is uniformly sampled from $\{1, \dots, H - h\}$, and H is the image height. In the experiments, h is set to 20.

5.6.1.7 Stochastic Corruption

We used two different masking processes. The first one denoted *Patch(n,k)* consists in selecting n patches of size $k \times k$ pixels. The top left position of each patch is uniformly sampled. These patches will correspond to an observation y, meaning that all the other non-selected pixels are set to 0. A small border of 4 pixels in the image is also excluded in order to keep the background consistent. The second one denoted *Drop Pixel* randomly samples a fraction p of pixels uniformly and the three corresponding channel values are set to 0.

5.6.1.8 Video Corruption

The above datasets provide ground truth videos without corruption. In order to generate corrupted observation sequences, we simulate different types of occlusion depending on the nature of the videos. Each corruption process is defined as a stochastic operator F as in Equation 5.22 with mask distribution

$p_{\mathbf{M}}$. For a given video one then generates a sequence of random masks, one mask being then associated to each frame of the sequence. Note that except for the Remove-Pixel corruption process where two successive corruptions are independent, for all processes, the generated corruption sequences are time-dependent: the corruption pattern at time t will depend on the one at time $t-1$.

Cloud This process is specific for the SST dataset. It simulates realistically video cloud masks on satellite images. Cloud masks are simulated using Liquid Water Path (LWP) data (measured in g/m²), which characterizes the total amount of liquid water present in the atmosphere between two points. The LWP data are generated by PyCLES (Pressel et al. 2015)[7], a large eddy simulation system. It simulates the evolution of clouds in time based on a variant of anelastic equations of the atmospheric motion. The binary masks are then obtained by setting the image pixels to 0 when their LWP value is above a threshold. This produces realistic cloud coverage of the captured regions, see Figure 5.16. Pixels occluded by the mask are set to $c = 1$. Thresholds are selected in the interval 55 to 80 g/m² to simulate clouds at different occlusion rates. Statistics about the occluded area at different thresholds are presented in Figure 5.20. For simulating occlusion, for each SST image sequence, we sample randomly a sequence of time coherent masks from the LWP dataset to be applied to the SST sequence.

Raindrops This process is a simplified model of random raindrops between subject and camera, taking into account a blurring effect when raindrops leave traces during exposure. It generates a set of white bars, each with a random length θ_l and a constant width w. Bars move down at a random speed θ_v, starting from a random initial position θ_p. All these values are normalized w.r.t the frame edge length in $]0,1[$. The number of raindrops is pre-defined. Bars return to the top once completely out of frame. Pixels occluded by the mask are reset to $c = 1$. Note that as for Cloud, this is a time-dependent measurement process, meaning that two successive masks are correlated.

Remove-Pixel This measurement roughly mimics severe damages on vintage films. It masks randomly a fixed proportion $p \in]0,1[$ of pixels at each time step and reset them to $c = 0$. Mask for each frame is generated independently regardless the evolution of time. This is the only time-independent measurement considered here.

Vertical-Moving-Bar This simple measurement operator generates a vertical white bar crossing the sequence, very roughly mimicking a fence or any similar

7. https://github.com/pressel/pycles

obstacle. The bar is generated with the following distribution parameters: width θ_w, initial position θ_p, horizontal constant velocity θ_v. These values are in $]0, 1[$ as for Raindrops. The moving direction is chosen randomly. The bar reappears on the opposite side once it reaches the border. Masked pixels in observation are reset to $c = 1$. This is a time-dependent measurement.

5.6.1.9 Evaluation Metrics

We used three complementary quantitative measures.

One could argue that the MSE is not a good metric for our model. Indeed, if for instance the input lossy measurement of a face is missing its eyes, the reconstructed eye color does not have any reason to be the same as the original color and the squared error is mechanically high. However, as we hope to output the most *plausible* possibility, the best model should have a low average MSE.

FID. The Frechet Inception Distance (Heusel et al. 2017) samples reconstructed examples $\tilde{x}$ and original uncorrupted examples x. The images are embedded into a feature space (the last layer of InceptionNet) where, the mean and the covariance of the embedding are estimated. The Frechet distance between these two statistics is then computed. The score is an indicator of the visual quality of the generated samples.

MSE or MAE. We compute the Mean Square Error (Mean Squared Error (MSE)) between the reconstruction and the original uncorrupted image. This is not a significant measure of the visual quality, but it provides an indicator of the reconstruction capacity of the models. The Mean Average Error is similar to the MSE but indicates the absolute deviation from the real data.

Standard Deviation To evaluate the diversity of the reconstructions, we compute the mean standard deviation of the reconstruction, over the mask reconstructed pixels, for 10 different samples z, over 1000 images. This correlates with the variation observed visually.

FVD Fréchet Video Distance (FVD, ("Towards Accurate Generative Models of Video" n.d.)), compare the activation distribution of the generated samples from $p_{\mathbf{X}}^{G}$ to the real one sampled from $p_{\mathbf{X}}$. These distributions are extracted from activation layers of NNs, which are pre-trained on video classification tasks. The two distances are calculated for the whole sequence including occluded and non-occluded regions.

5.6.2 Baselines

5.6.2.1 Deterministic Baselines

Conditional AmbientGan. The context is the same as for our model: the measurement process F is assumed known, there is no access to samples from the uncorrupted signal distribution p_X, but only to their corrupted counterpart p_Y.

An unconditional generator G is trained using the AmbientGan framework (Bora et al. 2018) for each type of measurement process F, in order to produce samples from p_X. The distribution induced by the generator p_X^G is an approximation of p_X (at the optimum, both distributions match, *i.e.* $\mathrm{p}_X^G = \mathrm{p}_X$). Given a specific measurement y, the reconstruction $\hat{x}$ is the signal from G that is closest to y, as in (Bora et al. 2017). To find $\hat{x} = G(\hat{z})$, we search for the latent code $\hat{z}$ of G, such that $\hat{z} = \arg\min_z \|y - G(z)\|_2^2 + R(z)$. $R(z)$ is a regularizing term that enforces the latent code to stay in G's input domain. This objective is optimized using stochastic gradient descent. To train G, we use the same architectures and hyperparameters as those provided by the authors. Because this approach may be sensitive to the initial latent code, we reiterate this approach three times and select the best resulting image.

Unpaired Variant. This is a variant of our deterministic model where we have access to samples of the signal distribution p_X. This means that although we have no paired samples from $\mathrm{p}_{X,Y}$, we have access to unpaired samples from p_X and p_Y. This baseline is similar to our model but instead of discriminating between a measurement from the data y and a simulated measurement $\hat{y}$, we directly discriminate between samples x from the signal distribution and the output of the reconstruction network $\hat{x}$.

Paired Variant. This is a variant of our model where we have access to signal measurement pairs (y, x) from the joint distribution $\mathrm{p}_{Y,X}$. Given input measurement y, the reconstruction is obtained by regressing y to the associated signal x using a MSE loss. In order to avoid blurry samples, we add an adversarial term in the objective in order to constrain G to produce realistic samples, as in (Isola et al. 2016). The model is trained using the same architectures as our model, and the hyperparameters have been found using cross-validation.

Deterministic Measurement Specific Baselines. We also compare our model to baselines that were designed to remove specific corruptions.

Deep Image Prior (Van Veen et al. 2018). Given an input measurement y, a generator G_ϕ parameterized by random parameters ϕ, and a random latent code

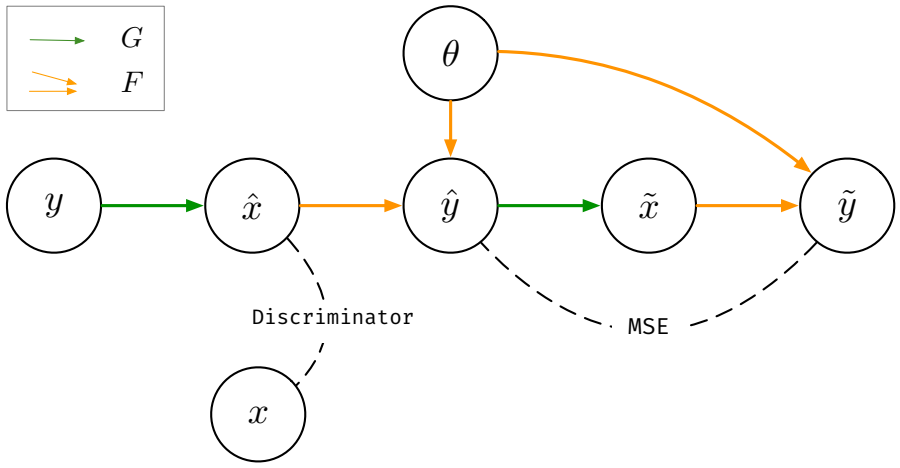


Figure 5.7 – Unpaired Variant of our model. As opposed to our deterministic model, this baseline has access to samples of the signal distribution p_X. This baseline is similar to our model, however, instead of discriminating between a measurement from the data y and a simulated measurement $\hat{y}$, we directly discriminate between samples from the signal distribution and the output of the reconstruction network $\hat{x}$.

z, the reconstruction $G_{\phi^*}(z)$ is obtained by resolving the following optimization problem:

$$\phi^* = \arg\min_{\phi} \left\| y - G_\phi(z) \right\|_2^2 \tag{5.29}$$

For measurement processes Patch-Band, Remove-Pixel and Remove-Pixel-Channel (refer to Section 5.6.1.6), the resulting reconstruction $G_{\phi^*}(z)$ was not satisfactory: G was consistently regressing to the corrupted values in the measurement y, which led to unsatisfactory results. Formally, instead of solving equation 5.29 we resolve the following objective:

$$\arg\min_{\phi} \left\| T(y - G_\phi(z); \theta) \right\|_2^2 \tag{5.30}$$

Where T acts as a mask, and eliminates the terms associated with the pixels from y that have been put to 0. Note that this method corresponds to the inpainting formulation in (Van Veen et al. 2018). We used the implementation provided by the authors [8].

Biharmonic Inpainting (Damelin et al. 2018). By considering inpainting as a smooth surface extension domain, this baseline resolves a biharmonic equation

8. https://dmitryulyanov.github.io/deep_image_prior

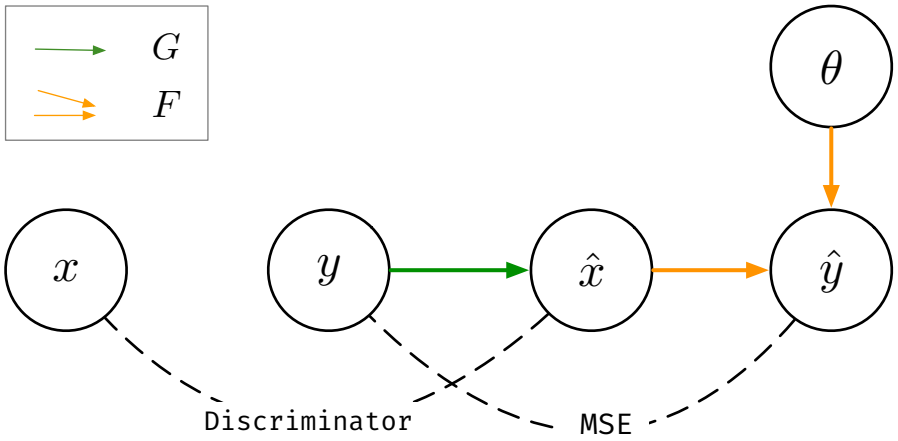


Figure 5.8 – Paired Variant of our model. As opposed to our deterministic model, this baseline not only has access to samples of the signal distribution p_X, but to signal measurement pairs (y, x) from the joint distribution $p_{Y,X}$. Given input measurement y, the reconstruction is obtained by regressing y to the associated signal x. In order to avoid blurry samples, we add an adversarial term in the objective in order to enforce G to produce realistic samples, as in (Isola et al. 2016). The model is trained using the same architectures as the ones from our model.

to obtain a high order approximation of the image. This approximation is then extended to the missing part of the image. This method assumes access to the θ associated to the observations in the data (*i.e.* in this case, the mask).

Total Variation Denoising (Chambolle 2004). This denoising baseline aims to minimize the total variation of an image i.e the integral of the absolute gradient of the image. Reducing the total variation of the image removes unwanted detail, such as white noise artifacts while preserving important details such as edges and corners.

5.6.2.2 Stochastic Baseline

For this setting, we also compared our model with several baselines. Note that in order to keep things comparable, all the networks used in the experiments are the same.

Unpaired Variant. This is a supervised variant of our model where we do not have access to paired (y, x) samples, but we have access to unpaired samples from p_X and p_Y. The baseline is similar to our model but instead of discriminating between a measurement from the data y and a simulated measurement $\hat{y}$, we directly discriminate between samples x from the signal distribution and the

output of the reconstruction network $\tilde{x}$. This should provide improved results w.r.t. the unsupervised model developed in the paper.

Paired Variant. Here we have access to corrupted-uncorrupted pairs (y, x) from the joint distribution $p_{Y,X}$. Given the masked image y, the reconstruction is obtained by regressing y to the associated complete image x using a MSE loss. In order to avoid blurry samples, we add an adversarial term in the objective, which helps G to produce realistic samples, as in (Isola et al. 2016).

MisGAN. This baseline is adapted from (Li et al. 2018). MisGAN (see the related work section) makes use of three generators; one for learning the data distribution, one for the mask distribution and one for inputting the data, In our adaptation, since we suppose the mask distribution known, we replace the corresponding component in their model with the true corruption process as in our model.

5.6.2.3 Temporal Baselines

Unsupervised Approaches We use two unsupervised baselines, one adapted for SST and the other one specific of natural videos. The former is DINEOF (Alvera Azcarate et al. 2005). This is a state-of-the-art data-driven completion method in geophysics, and it has been used for SST observations, chlorophyll, salinity, etc. It is a parameter-free interpolation technique based on empirical orthogonal function (EOF). It adopts an iterative algorithm that calculates at each iteration a truncated decomposition of EOF from known pixels, then replaces the values marked as missing by a reconstruction from calculated EOF. DINEOF does not make any assumption on the form of missing area and as such could be used for other domains as well and for different types of complex occlusion processes. However, DINEOF has been developed for remote sensing and does not ensure the coherence between different input channels (e.g. for RGB images).

The other one is Newson et al. 2014, one of the very few methods for unsupervised natural video inpainting. It is representative of patch-based approaches and it is still today state-of-the-art for many natural video occlusion processes. It searches for the nearest neighbors of occluded area using an Approximate Nearest Neighbor (ANN) search. The occluded area is reconstructed by assembling information from these neighbors at multiple scales. The form of the researched patches is supposed to be rectangular cuboids, e.g. a $5 \times 5 \times 5$ spatiotemporal tensor, which limits its capability to adapt to more complex cases like Cloud, Raindrops, Remove-Pixel.

Supervised Approaches There exists several supervised approaches to sequence inpainting. In order to evaluate the performance of our unsupervised

method w.r.t. supervised ones, we compared with two supervised baselines. As our goal is not to beat state-of-the-art supervised techniques, we used two supervised adaptations of our model, respectively trained using unpaired and paired supervision. They are described below.

UNPAIRED VARIANT This is a supervised variant of our video model in which we have access to unpaired samples from $p_{\mathbf{X}}$ and $p_{\mathbf{Y}}$ because we have access to clean $\boldsymbol{x}$ data, it is then possible to supervise the approximation $\hat{\boldsymbol{x}} = G(\boldsymbol{y})$ by discriminating directly between samples $\boldsymbol{x}$ from the signal distribution and the output of the reconstruction network $\hat{\boldsymbol{x}}$. This is similar to the unpaired model of the deterministic image model. PAIRED VARIANT Here we have access to corrupted-uncorrupted pairs $(\boldsymbol{y}, \boldsymbol{x})$ from the joint distribution $p_{\mathbf{Y},\mathbf{X}}$. Given the masked image $\boldsymbol{y}$, the reconstruction is obtained by regressing $\boldsymbol{y}$ to the associated complete image $\boldsymbol{x}$ using a L^1 loss. In order to avoid blurry samples, we add an adversarial term in the objective, which helps G to produce realistic samples. This model is similar to the Vid2Vid (T.-C. Wang et al. 2018) model, except that they rely on optical flow which is not available in our case because of the masked regions. This is similar to the paired model of the deterministic image model.

5.6.3 Deterministic Results

We will now present our results. First, we compare quantitatively our model with non-measurement specific baselines on CelebA. We then present qualitative results with samples from our model and these baselines. Comparisons with measurement specific baselines are presented in Section 5.8 for the three datasets.

5.6.3.1 Quantitative Results

We compare our model with the baselines introduced in the previous section. We report MSE scores between the reconstructed $\hat{x}$ and the true signal x used to generate the input y. Table 5.1 shows the MSE computed on the *test* set, a randomly selected subset of CelebA comprised of 40000 images. Because the Conditional AmbientGan model is too computationally expensive, we only report the MSE on 40 randomly chosen samples of the test set.

Quantitatively, our model performs well. Except for the Conditional AmbientGAN, all the methods are quite similar in terms of MSE. Our unsupervised model reaches performance similar to its variants trained using additional supervision. We also note that when the aligned signal-observation pairs are not used (as in our Unpaired Variant), results are comparable – sometimes better – than when these pairs are used directly (as in our Paired Variant). This suggests

Table 5.1 – Average MSE of neural network based models on the test set of CelebA, for different measurement processes. The first two rows are models trained with no supervision, the last two rows with additional supervision.

	Remove-Pixel	Remove-Pixel-Channel	Patch-Band	Convolve-Noise
Conditional AmbientGan	0.292	0.2829	0.1421	0.0814
Our Deterministic Model	**0.0414**	**0.0409**	**0.0165**	**0.0088**
Unpaired Variant	**0.037**	**0.0336**	0.034	0.0103
Paired Variant	0.0383	0.0401	**0.0147**	**0.0084**

that our likelihood term is sufficient to condition the reconstruction on the input signal.

5.6.3.2 Qualitative Results

We now evaluate the quality of our reconstruction on three different datasets (Section Section 5.6.1). Figure 5.9 shows reconstructions obtained from different models on the CelebA dataset. As we can see, the measurement functions we used, induce a large loss of information, and is difficult to reconstruct, even for a human. We observe that Conditional AmbientGAN yields visually poor results, especially for the Remove-Pixel and Remove-Pixel-Channel measurement processes. We hypothesize that this is due to the large Euclidean distance between the measurements and the associated signals, and the suboptimality of the generator. Visually, the quality of our model's reconstructions are coherent with the quantitative results: they are comparable to its paired and unpaired counterparts (Section 5.6.2). Figures (5.10), (5.11), (5.12), and (5.13) each show reconstructions from a given measurement process on different datasets. Our model is able to produce images with good visual quality while remaining coherent with the underlying uncorrupted images. In Figures (5.28), (5.29), (5.30) and (5.31) in Section 5.8, we compare our model with commonly used inpainting or denoising methods. We can see that contrary to these methods, we are able to capture semantic information from the dataset. Typically, in Figure 5.31, the model infers missing eyes or noses, without ever having seen them. Additional samples are available in Section 5.8, refer to Figures (5.32), (5.33), (5.34), and (5.35).

5.6.4 Stochastic Results

We first provide some examples of the images generated by the model presented in Section 5.4 for visual inspection and then detail quantitative results comparing our model to different baselines.

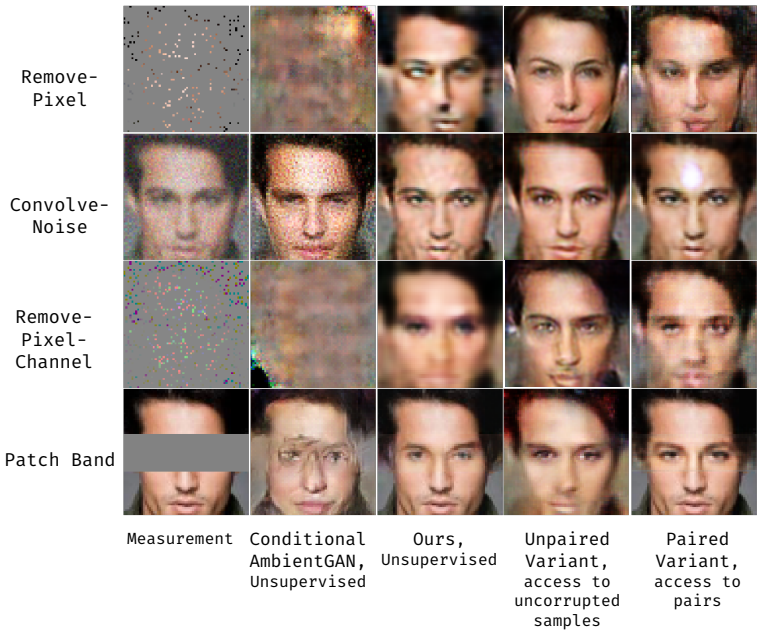


Figure 5.9 – Model reconstructions for different corruption processes, on CelebA. Each row corresponds to a specific corruption process, and each column to a particular model.

Figure 5.10 – On the top row, randomly sampled test set measurements from CelebA corrupted using Patch-Band($h = 20$), and below, our associated reconstructions.

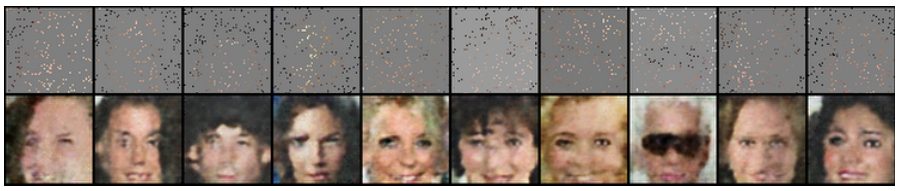


Figure 5.11 – On the top row, randomly sampled test set measurements from CelebA corrupted using Remove-Pixel($p = 0.95$), and below, our associated reconstructions.

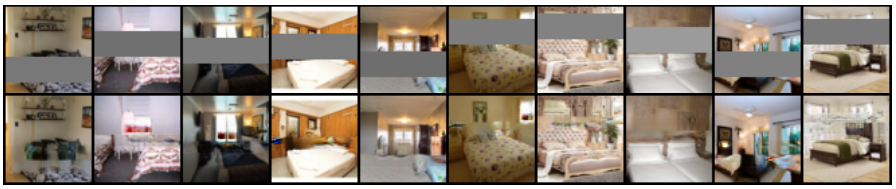


Figure 5.12 – On the top row, randomly sampled test set measurements from LSUN corrupted using Patch-Band($h = 20$), and below, our associated reconstructions.

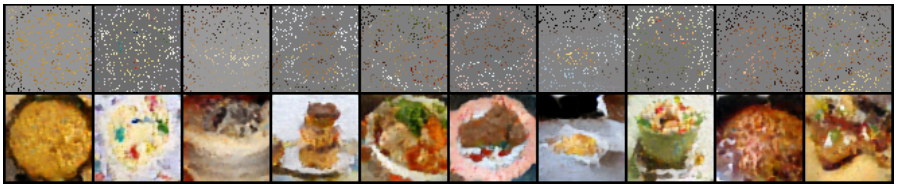


Figure 5.13 – On the top row, randomly sampled test set measurements from Recipe-1M corrupted using Remove-Pixel ($p = 0.9$), and below, our associated reconstructions.

5.6.4.1 Qualitative evaluation

Figure 5.14 and 5.15 provide examples of reconstructions obtained on the CelebA dataset with our model (Equation 5.21), respectively for the *Patch* and for the *Drop Pixel* corruptions. The results are obtained using the model in Figure 5.4. For each figure, the top row is the observation y and the rows below are the associated reconstructions for different samples $z \sim p_z$. For these experiments, the settings are respectively *Patch(90,10)*, i.e. 90 patches are selected from a full image x, each of size 10 pixels to build observation y, all the other values being set to 0, and *Drop Pixel*, where 90% of the pixels are randomly selected and their values on the three channels set to 0.

For each experiment, a specific training was performed for the selected corruption model, meaning that a training was performed for Figure 5.14 and another one for Figure 5.15. As can be seen, the reconstruction is not perfect, but given the large amount of corruption and the unsupervised setting, the model is able to reconstruct a large part of the information present in the original image, together with a significant diversity. The latter could be seen by inspecting the two figures 5.14 and 5.15. For example, the eyes most often come in different shapes and colors. Both figures exhibit random - not cherry-picked - samples. The reconstruction quality depends on the noise nature. It is easier for the *DropPixel* corruption than for the *Patch* one which is particularly difficult.

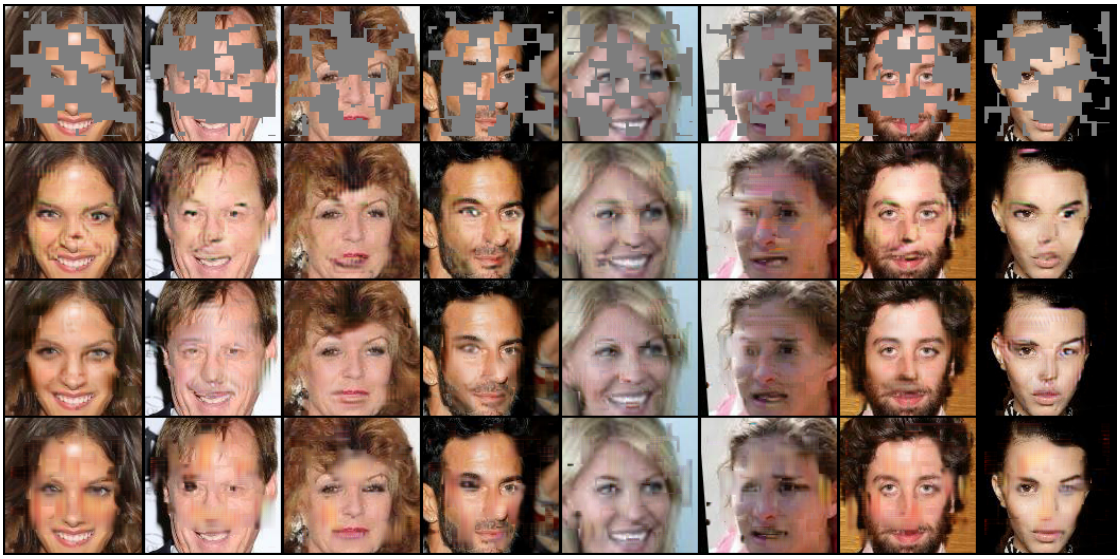


Figure 5.14 – On the top row, randomly sampled measurements, from CelebA corrupted using *Patch* with $n = 90$ and $k = 10$, and below associated reconstructions, for different latent vector z.

5.6.4.2 Quantitative analysis

Ablation Study In order to analyze the importance of the different model components, we performed an ablation analysis, by training the model without any auxiliary loss (equation 5.18), with the *Encoding* z auxiliary loss (equation 5.19), with the *Encoding* y auxiliary loss (equation 5.20) and with both losses used together (equation 5.21). Results are reported in table 5.2 for the CelebA 128×128 dataset for three corruption processes: *Patch(1,32)* corresponds to a single patch of size 32×32 for the observation y, *Patch(90,10)* corresponds to 90 small patches of size 10 pixels for y, *Drop Pixel* corresponds to 90% of pixels selected with their three-channel values set to zero. In table 5.2 we report the values for the three quantitative measures: MSE scores between the reconstructed $\tilde{x}$ and the true signal x used to generate the input y, Frechet Inception Distance (FID) and the standard error deviation (see section 5.6.1.9).

Table 5.2 – MSE, FID, and standard deviation on the celebA dataset for three noises.

		One Patch			Small Patch			Drop Pixel		
Encoding z	Encoding y	FID	MSE	std.	FID	MSE	std.	FID	MSE	std.
o	✓	59.39	0.159	0.0463	20.37	0.062	0.024	69.73	0.130	0.058
✓	o	56.38	0.161	0.0408	26.39	0.062	0.024	**63.07**	**0.089**	0.074
o	o	**53.89**	**0.149**	0.0325	28.42	0.069	0.014	74.82	0.131	0.026
✓	✓	54.55	0.156	**0.0645**	**19.11**	**0.059**	**0.035**	70.19	0.130	**0.109**

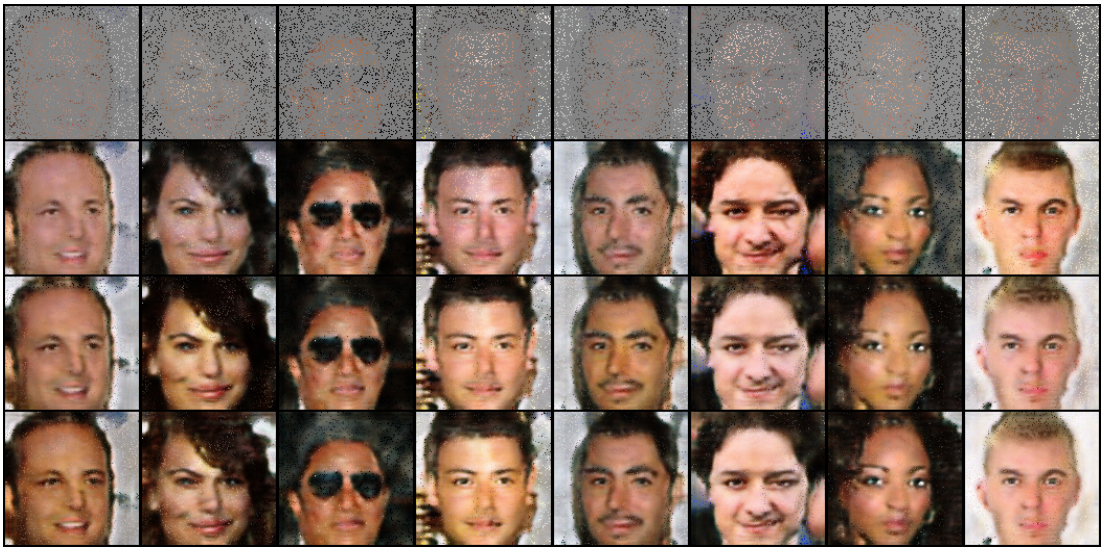


Figure 5.15 – On the top row, randomly sampled measurements, from CelebA corrupted using *DropPixel* with $p = 0.90$, and below associated reconstructions, for different latent vector z.

The three models implementing an auxiliary loss, all improve the diversity for the three corruption processes (*std.* column) w.r.t. the simple base model (Equation 5.18). The model combining the two auxiliary losses is clearly better than the others for sample diversity. It thus seems better suited for learning to generate reconstructed image distributions. For the large patch experiment, all the models exhibit very close performance, while the simp;le base model without any auxiliary loss performs slightly better. Here diversity comes at the expense of the reconstruction fidelity as measured by the FID and MSE criteria. For the multiple patches *Patch(90,10)* experiments, the models with auxiliary loss all increase the MSE and FID performance w.r.t. the simple model (equation 5.18), with the combined auxiliary loss clearly better than the other variants. For the *Drop Pixel* experiment, again the proposed methods improve over the simple model, but here the *Encoding z* version is better than the others.

Results in Table 5.2 are coherent with the ones reported in (Zhu et al. 2017b). As the reconstruction performance are of the same magnitude, adding the Latent Reconstruction losses improve the diversity of the model.

Table 5.3 – MSE, FID, and standard deviation on the celebA dataset for three noises and different baselines.

Model	One Patch			Small Patch			Drop Pixel		
	FID	MSE	std.	FID	MSE	std.	FID	MSE	std.
Unpaired	46.90	0.129	0.0359	18.55	0.053	0.015	60.28	0.098	0.032
Paired	**45.66**	**0.113**	-	**18.32**	**0.044**	-	**59.38**	**0.078**	-
Misgan	84.63	0.166	0.0322	25.37	0.101	0.014	86.42	0.149	0.027
Our Model	54.55	0.156	**0.0645**	19.11	0.059	**0.035**	70.19	0.130	**0.109**

Comparison with baselines As shown in table 5.3, our unsupervised model (shown here with the combined auxiliary losses) reaches performance close to its variants trained using additional supervision. MisGAN shows slightly worse performance, due to the need to minimize the Generator and the Imputer GAN Losses. The convergence of this model is also order of magnitude slower than ours. As before, the proposed model largely increases the diversity compared to the baselines meaning that it better learns a distribution of the reconstructed examples. MisGAN also makes use of a stochastic input as our model does but it is unable to generate examples with a significant diversity.

5.6.5 Temporal Results

Figure 5.16 – SST

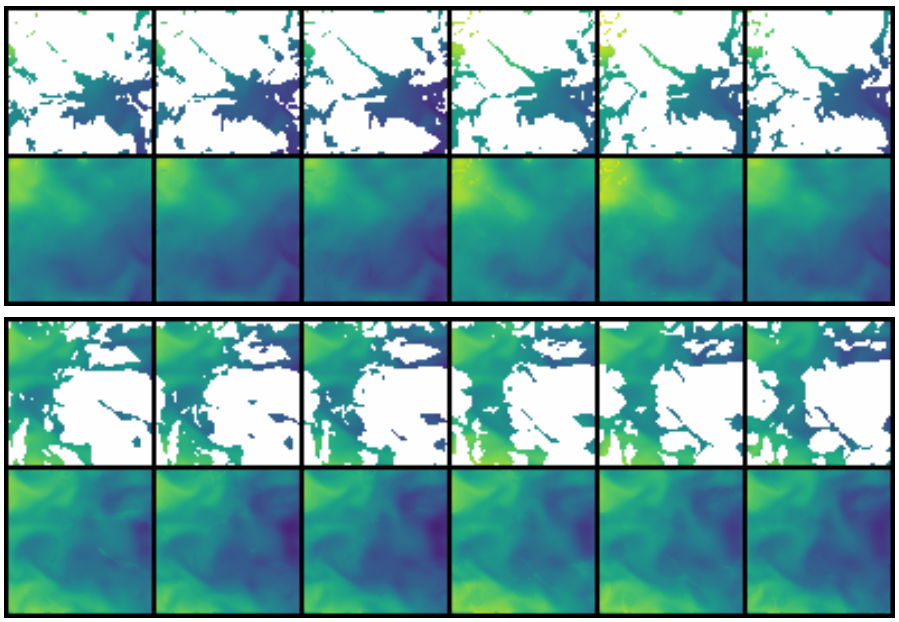


Figure 5.17 – FaceForensics++

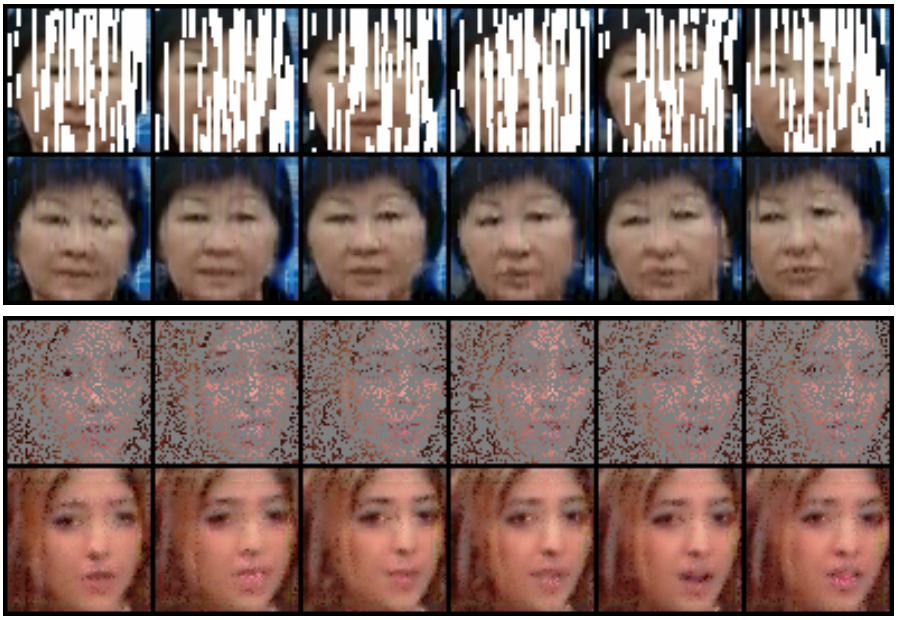
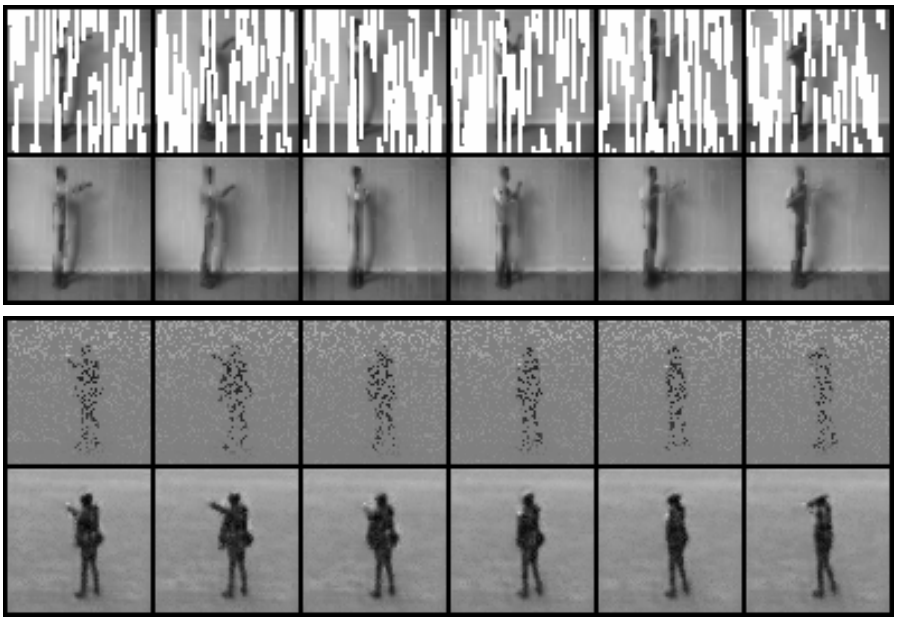
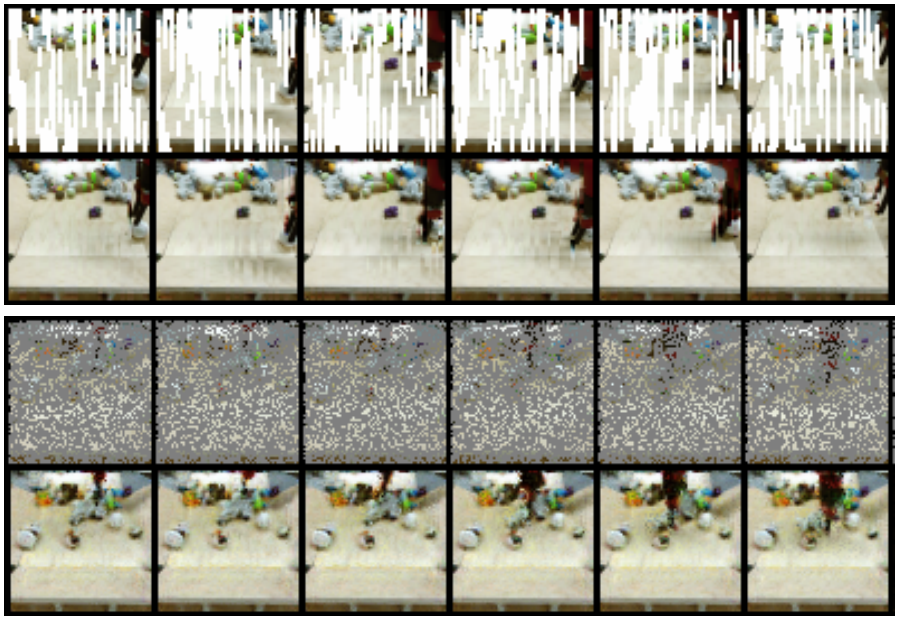


Figure 5.18 – KTH



We show here samples from test sets. SST data (5.16) are masked with Cloud, natural video datasets (firuges 5.17,5.18,5.19) are masked with Remove-Pixel and Raindrops. Sequences are accelerated 3 times to make movements more visible. For each figures, row 1 and 3 correspond to observation y and rows 2 and 4 to their respective reconstruction $\hat{x}$.

Figure 5.19 – BAIR



5.6.6 Comparison with Baselines

LWP (g/m²)	Occluded Area (%)	FID	FVD	MAE (°C)
55	79.9± 9.6	32.49	134.40	.1273±.0443
60	69.6±12.8	22.95	79.13	.1047±.0396
65	55.9±15.1	17.75	75.07	.0988±.0378
70	39.5±14.6	8.01	40.76	.0739±.0324
75	24.5±11.5	5.58	30.07	.0698±.0305
80	13.4± 7.8	1.77	9.89	.0497±.0237
All	47.1±11.9	14.76	61.55	.0874±.0347

Figure 5.20 – Results with clouds generated at different LWP thresholds.

Method	FID	FVD	MAE (°C)
Ours	**8.01**	**40.76**	**.0739±.0324**
Alvera Azcarate et al. 2005	27.99	323.61	.1214±.0248
Newson et al. 2014	—*	—*	—*

Figure 5.21 – Comparison of results with clouds at LWP threshold 70 g/m². *Unable to finish.

Results for SST Data Figure 5.20 shows the results for SST data with simulated clouds at different occlusion rates. For most occlusion rates, the generated

Dataset	Method	Raindrops			Remove-Pixel			Vertical-Moving-Bar		
		FID	FVD	MAE	FID	FVD	MAE	FID	FVD	MAE
FF++	Ours	**43.72**	**1574.89**	**.0834±.0187**	**93.28**	**1460.02**	.0894±.0137	19.12	493.57	.1304±.0972
	(1)	75.93	3424.11	.1208±.0272	110.15	3091.67	**.0752±.0161**	56.58	5775.25	.3286±.0815
	(2)	—*	—*	—*	—*	—*	—*	**9.04**	**316.55**	**.0494±.0501**
KTH	Ours	**56.56**	**2522.81**	**.0380±.0062**	**56.16**	**2639.24**	.0429±.0037	39.05	588.94	.0711±.0505
	(1)	71.69	6400.44	.0522±.0073	82.45	6660.02	**.0403±.0040**	34.90	3408.19	.0959±.0402
	(2)	—*	—*	—*	—*	—*	—*	**11.88**	**354.01**	**.0268±.0403**
BAIR	Ours	**27.33**	**1194.19**	**.0821±.0153**	**53.80**	**2073.90**	**.0997±.0087**	11.55	496.38	.1619±.0590
	(1)	89.87	4456.08	.2345±.0274	140.20	4014.17	.1424±.0103	67.06	7361.77	.5579±.0766
	(2)	—*	—*	—*	—*	—*	—*	**10.31**	**340.97**	**.1082±.0873**

Table 5.4 – Results for FaceForensics, KTH, and BAIR. Compared with (1) (Alvera Azcarate et al. 2005) and (2) (Newson et al. 2014). *Unable to finish.

sequences have an Mean Average Error (MAE) under 0.1 °C which is well below the reference baseline (see Figure 5.20). They also have good FID and FVD values, which means that they are spatially and temporally realistic (See Figure 5.16 for examples). For heavily occluded area, our model can realistically reconstruct the data around the border, while the reconstruction near the center of the cloud is of lower quality. We compare our results in Figure 5.21 at 70% occlusion, with DINEOF, the SOTA agnostic method for image reconstruction in IR images. The error reduction w.r.t. DINEOF is about 40% for MAE. We have not been able to obtain results for (Newson et al. 2014) in reasonable time for such complex masks. Note that (Newson et al. 2014) specifically designed for imputation in natural videos is not adapted for this type of occlusion.

Results for Videos Table 5.4 gathers the results obtained for the three natural video datasets with artificial measurements (Raindrops, Remove-Pixel, and Vertical-Moving-Bar). For all measurements, the FID and Frechet Video Distance (FVD) performance obtained by our model are 20%-50% better than DINEOF. This means that our model better controls both the spatial and the temporal generation quality than DINEOF. Globally, we achieve better MAE scores notably for color videos with few exceptions (performance are close for Remove-Pixel). As for Newson et al. 2014, the calculation could not be terminated in a reasonable time for highly complex measurements such as Raindrops, and Remove-Pixel, which make their search for cuboid patches in non-occluded area extremely hard. Newson et al. 2014 performs better when the form of the masks is simple such as the Vertical-Moving-Bars, for which completing patches could be easily found in neighbor frames. However, the computation time of Newson et al. 2014 is much higher than our model. Note that reduced computation time was an argument put forward in their publication. For a 30 frames 64×64 video,

Newson et al. 2014 costs on average 1 minute, versus around 1 second by our model.

Method	FID	FVD	MAE
Ours, Unsupervised	43.72	1574.89	.0834±.0187
Unpaired, Supervised	**20.86**	**575.08**	**.0547±.0105**
Paired, Supervised	22.17	720.75	.0555±.0108

Figure 5.22 – Comparison with supervised baselines for FaceForensics++ with Raindrops.

Comparison with Supervised Baselines Figure 5.22 compares our model with the two supervised (unpaired and paired) variants described in Section 5.6.2. Unsurprisingly, the performance of supervised models is far better than the ones of our unsupervised model. We find out that the access to the ground truth reduces dramatically the results for all three metrics. By using supervision, FID is halved and FVD is between two and three times smaller. The error reduction is smaller with MAE. We also notice that the unpaired version performs better than the paired one in terms of sequence completion quality (FVD) as the L^1 loss introduces a strong constraint for the reconstruction. It might be that our task is close to a generation task, and that the L^1 loss constraints too much the generation.

5.6.7 Ablation Study

Method	FID	FVD	MAE (°C)
Ours	**8.01**	**40.76**	**.0739±.0324**
Recurrent variant	13.29	67.37	.0960±.0431
Static variant	35.91	279.78	.1036±.0047

Figure 5.23 – Comparison of results for SST data for ablation study

We also conduct additional experiments in order to quantify the importance of the temporal component.

In a first series of experiments, we remove the sequence component from our model, i.e. removing the sequence discriminator D_s and replacing the 3D generator by a 2D one generating individual frames.

Figure 5.23 shows that our model clearly improves temporal quality by reducing FVD by a ratio of 7 compared to the model without the temporal component (denoted Static variant in the table). Note that FID is also clearly improved by a factor of 4. This gives more evidence that the model is able to exploit temporal dependency for its image completion task. We provide samples for this part in Figure 5.27.

Figure 5.24 – Pajot et al. 2018. Abrupt inter-frame changes degrade temporal quality

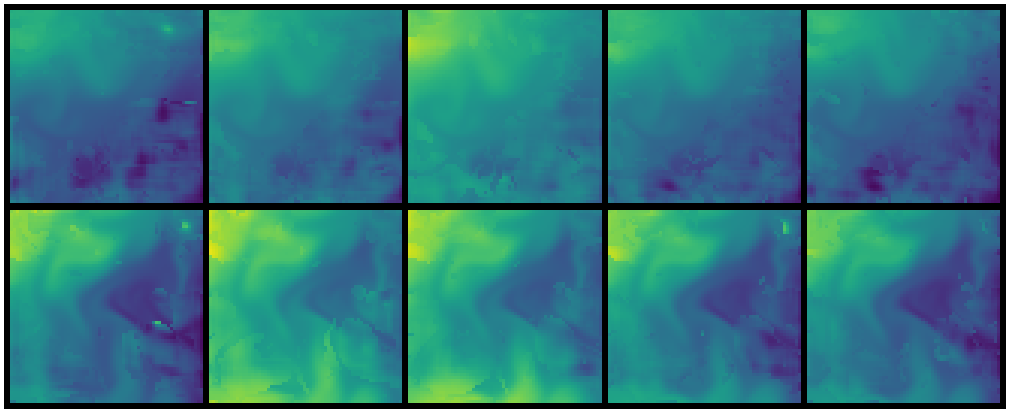


Figure 5.25 – Recurrent variant. Frames on the right are better than those on the left.

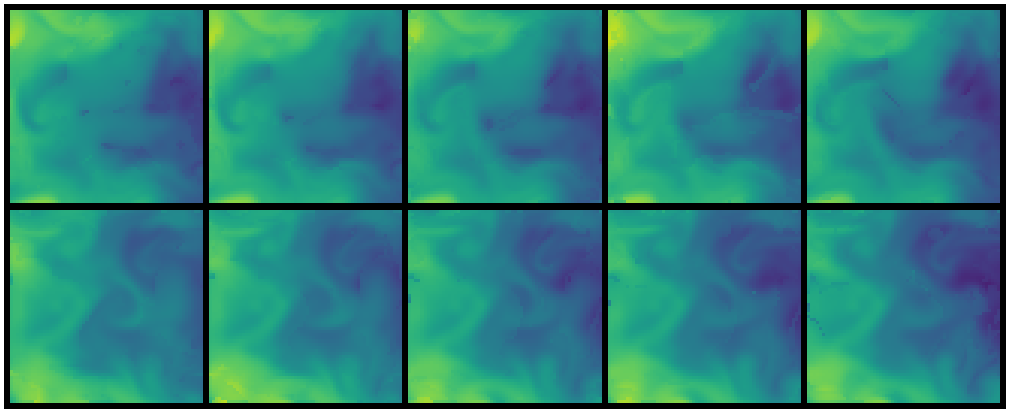


Figure 5.26 – Ours

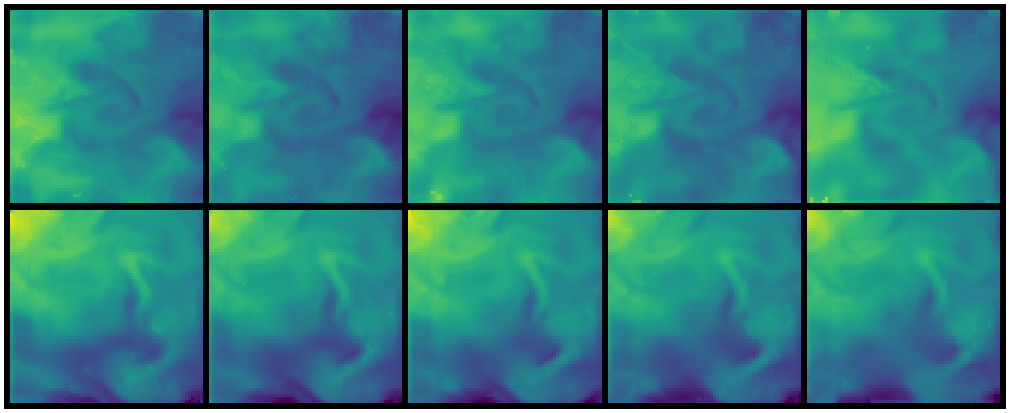


Figure 5.27 – Samples for ablation study.

Our model generates a frame at time t, $\hat{\boldsymbol{x}}_t$ from a whole sequence of observations $\boldsymbol{y}$. In a second series of experiments, we conditioned the generation of frames $\hat{\boldsymbol{x}}_t$ only on past observations. We feed past observations into a convolutional Recurrent Neural Network (RNN) (we used Gated Recurrent Unit (GRU) in our experiments) and generate the reconstructed frame, still denoted $G(\boldsymbol{y})$ by abuse of notation, from the last hidden state of the RNN, which encodes all past observations. The spatial discriminator operates as before, while the sequence discriminator operates on past observations only, instead of the full sequence of observations in our model.

See Section 5.6.2.3 for an illustration and for further description. Results in Figure 5.23 - Recurrent variant, show that using only past observations makes the completion less realistic and less accurate, but it still clearly outperforms the model without time dependency.

5.7 Conclusion

We have proposed a general formulation to recover a signal from lossy measurements using neural networks, without having access to uncorrupted signal data. We have formulated the problem as finding a *maximum a posteriori* estimate of the signal given its observation, for all observations in the training set. This gives us a natural objective for our neural network, composed of a linear combination of an adversarial loss for recovering realistic signals, and of a reconstruction loss to tie the reconstruction to its associated observation. Our approach yields results superior to the baselines, while staying competitive with other model variants that have access to higher forms of supervision.

We have also proposed two variants. One where we present a new formulation for learning the distribution of inpainted images in an unsupervised setting, where only incomplete observations are available. The problem has been formulated using a Conditional Generative Adversarial Network setting. Training relies on the optimization of a criterion combining an adversarial loss allowing the recovery of sharp realistic signals, and a latent reconstruction loss allowing to condition the reconstructed image to an incomplete observation. For the second variant, we tackle the problem of inpainting a sequence of occluded observations. Our model is augmented with two discriminators classifying real and generated observation sequences. We show that our model is able to complete spatiotemporal data without ground truth supervision when we have a stochastic model of the occlusion process. Our results for SST data and natural videos show that the recovered sequences are realistic, especially when the occluded area is highly complex.

5.8 Additional Samples

Figure 5.28 – Baseline comparison for the CelebA dataset. Corruption is Remove-Pixel ($p = 0.95$).

Figure 5.29 – Baseline comparison for the CelebA dataset. Corruption is Remove-Pixel-Channel ($p = 0.95$).

Figure 5.30 – Baseline comparison for the CelebA dataset. Corruption is Convnoise($\sigma_C = 0.2$, $l = 3$).

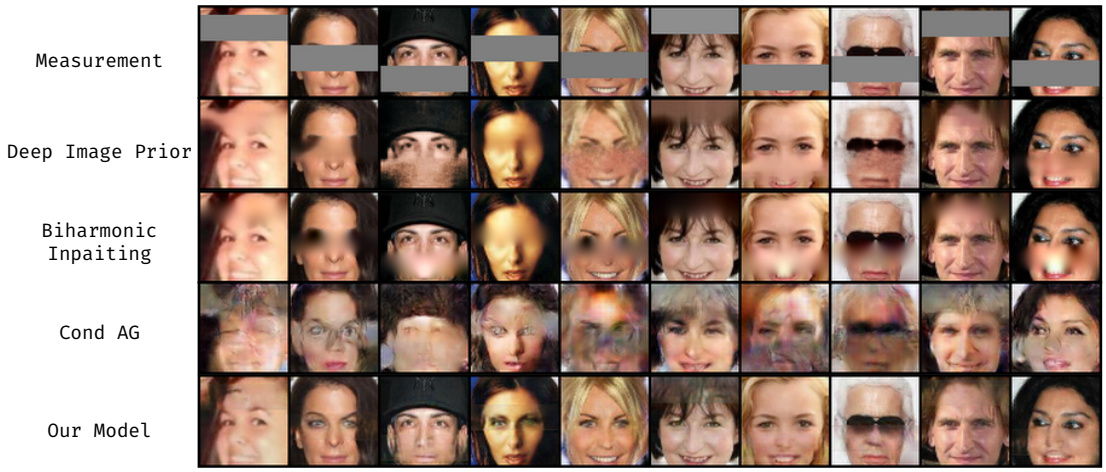


Figure 5.31 – Baseline comparison for the CelebA dataset. Corruption is PatchBand($h = 20$).

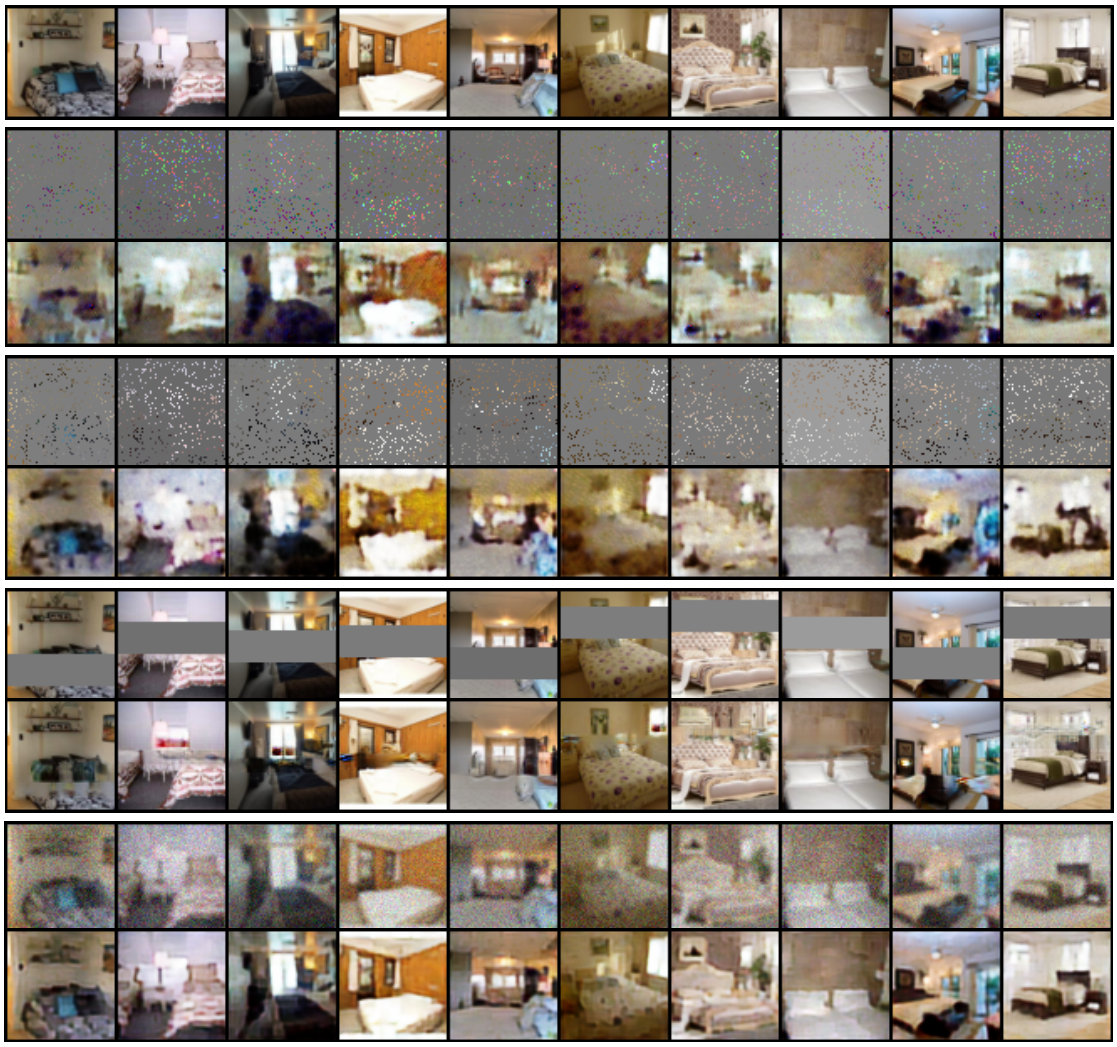


Figure 5.32 – On the top row, randomly sampled test set images from LSUN. Below, associated couples of corrupted observations and subsequent reconstructions from our model. From top to bottom, corruptions are Remove-Pixel-Channel($p = 0.95$), Remove-Pixel($p = 0.90$), Patch-Band($h = 20$), Convnoise($\sigma_C = 0.15$, $l = 3$).

Figure 5.33 – On the top row, randomly sampled test set images from Recipe. Below, associated couples of corrupted observations and subsequent reconstructions from our model. From top to bottom, corruptions are Remove-Pixel-Channel($p = 0.95$), Remove-Pixel($p = 0.90$), Patch-Band($h = 20$), Convnoise($\sigma_C = 0.15$, $l = 3$).

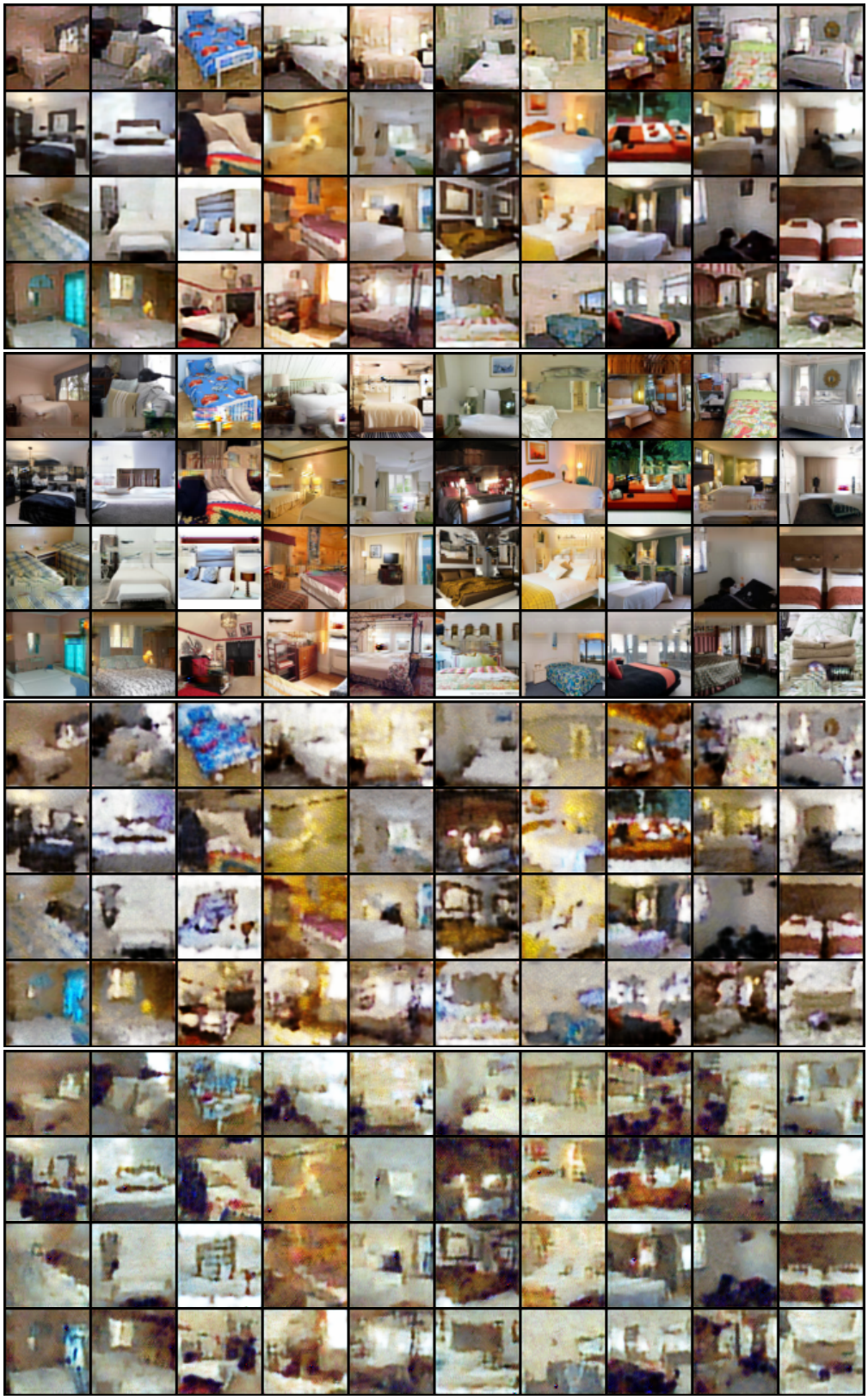


Figure 5.34 – Additional samples from our model of the LSUN Bedrooms dataset. From top to bottom, corruptions are Convnoise($\sigma_C = 0.15$, $l = 3$), Patch-Band($h = 20$), Remove-Pixel-Channel($p = 0.90$) and Remove-Pixel($p = 0.95$).

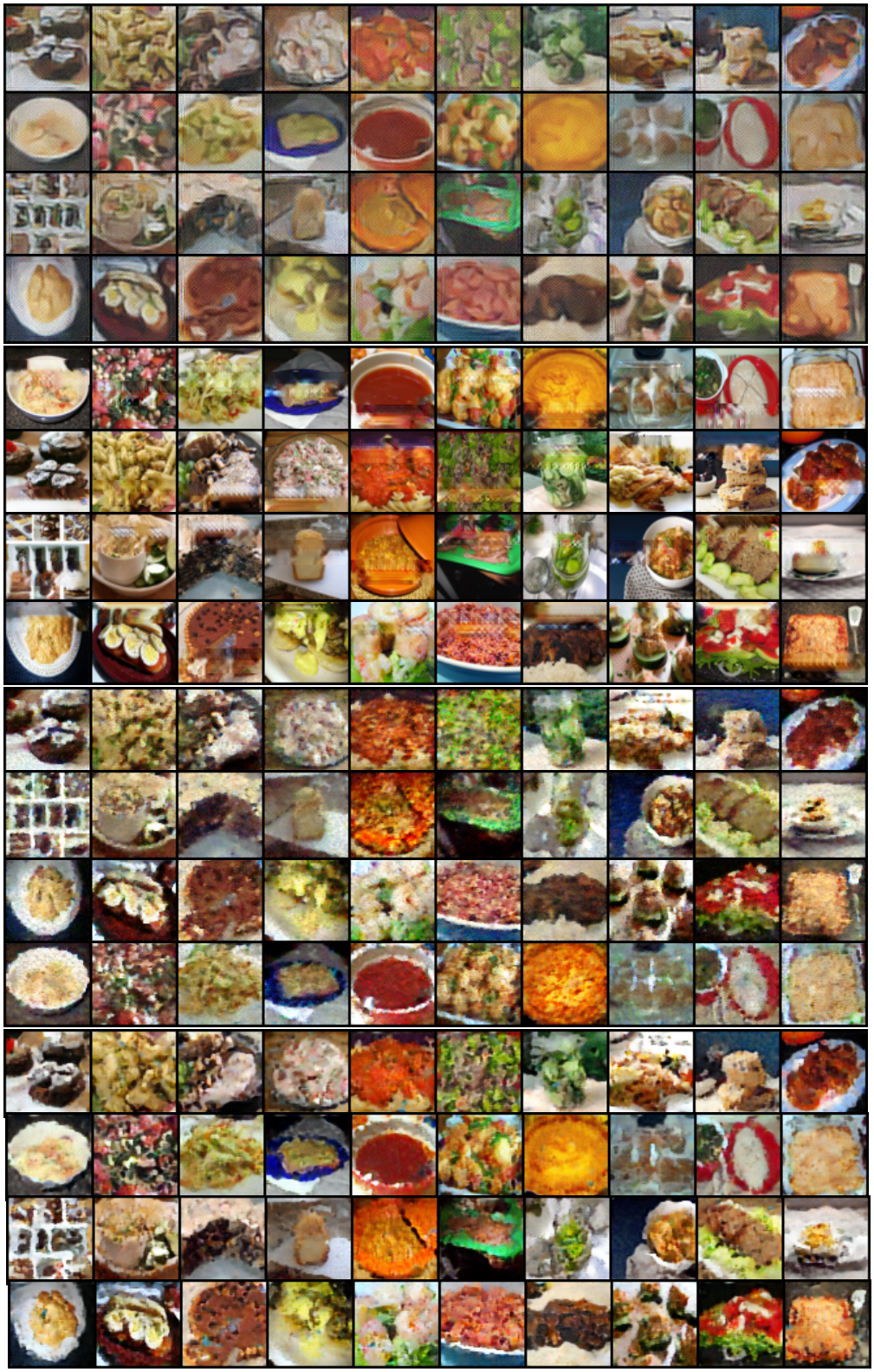


Figure 5.35 – Additional sample from our model, on the Recipe dataset. From top to bottom, corruptions are Convnoise($\sigma_C = 0.3$, $l = 5$), Patch-Band($h = 20$), Remove-Pixel-Channel($p = 0.90$) and Remove-Pixel($p = 0.90$).

Figure 5.36 – On the top row, randomly sampled measurements, from the LSUN dataset corrupted using *Patch* with $n = 90$ and $k = 10$, and below associated reconstructions, for different latent vector. z

Figure 5.37 – On the top row, randomly sampled measurements, from the Recipe dataset corrupted using *Patch* with $n = 90$ and $k = 10$, and below associated reconstructions, for different latent vector. z

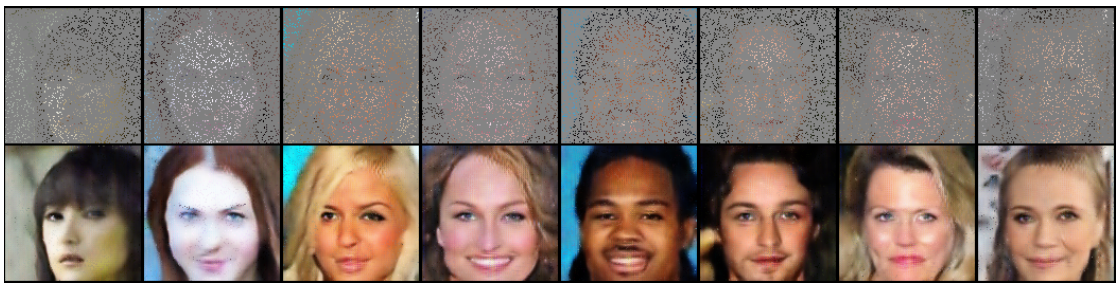


Figure 5.38 – On the top row, randomly sampled measurements, from the CelebA dataset corrupted using *DropPixel* with $p = 0.90$, and below associated reconstructions from the *paired* baseline.

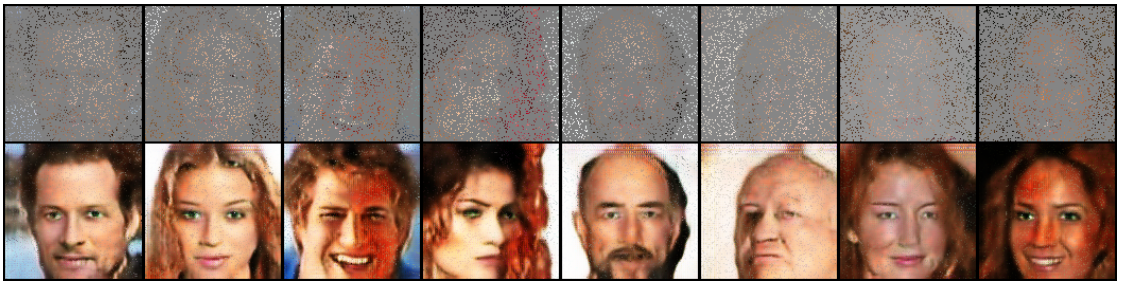


Figure 5.39 – On the top row, randomly sampled measurements, from the CelebA dataset corrupted using *DropPixel* with $p = 0.90$, and below associated reconstructions from the *unpaired* baseline.

Chapter 6

CONCLUSION

In this thesis, we have presented our contribution for using Machine Learning, and more specifically Deep Learning, to model dynamical system. Dynamical system benefit from two paradigms : a physical paradigm which is based on the knowledge from physicist and domain specialist, and a statistical one, which is based on data. We tried to explicit both paradigms by building hybrid dynamical systems.We focused on three tasks : forecasting, hidden state discovery, and inverse problem solving. In the following we highlight our contributions, as future work emerging from this thesis.

6.1 Summary of Contributions

Incorporating Prior Knowledge in Forecasting spatiotemporal data

In Chapter 3, we have shown that using a data-intensive framework offers a realistic alternative to the more classical physical approaches for modeling complex natural processes. We believe that using both the statistical and physical paradigms is essential for efficient modeling of complex physical phenomena. By using as an example application a problem of intermediate complexity concerning ocean dynamics, namely SST forecasting, we proposed a Deep Learning models using inspiration from the physics under the form of Partial Differential Equation (PDE). By using a solution of the advection-diffusion equation, our model contains two modules: an estimator that, given some input SST frame, can estimate the underlying flow and a Gaussian advection module that advect the last image with the estimated flow. As we explicitly model the displacement, we propose to add physical regularization, such as on the divergence or the gradient of the flow, to the loss function. The proposed approach can be easily generalized to a class of problems for which the underlying dynamics follow advection-diffusion principles. We have compared the proposed approach to a series of baselines. It is able to reach performance comparable to a state-of-the-art numerical model and clearly outperforms alternative models used as baselines.

Learning Dynamical Systems from Partial Observations

In Chapter 4, we have introduced a general data-driven framework to predict the evolution of spatiotemporal processes, when the system is complex and possibly nonlinear and the state is not fully observed. Assuming the underlying system follows a time-dependent differential equation, we estimate the unknown evolution term with a neural network. We argue that this is a natural way to model continuous-time systems. We propose a learning algorithm for the neural network, and two models : one where we estimate the initial state variables from some initial condition and one where we estimate the initial state variables from the observations. We then place the forecast neural network in an Ordinary Differential Equation solver in order to produce future predictions. Experiments performed on two simulated datasets from fluid dynamics and on data from a sophisticated data simulator used in climate modeling shows that the proposed method not only is able to produce high quality forecasts at different horizons, but also learns with a good accuracy the underlying state space dynamics. We have also proposed an experiment where we interpolate in time a physical system with our neural network, which shows that the model has learned the true dynamic.

Unsupervised Image Reconstruction

In Chapter 5 we have proposed a new formulation for learning the distribution of a signal using lossy measurements in an unsupervised setting, where only incomplete observations are available. The problem has been formulated using a Conditional Generative Adversarial Network Generative Adversarial Network (GAN) setting. Training relies on the optimization of a criterion combining an adversarial loss allowing the recovery of sharp realistic signals, and a latent reconstruction loss allowing conditioning the reconstructed image to an incomplete observation.

We have formulated the problem as finding a *maximum a posteriori* estimate of the signal given its observation, for all observations in the training set. Our approach yields results superior to the baselines, while staying competitive with other model variants that have access to higher forms of supervision.

We have also proposed two variants. One where we present a new formulation for learning the distribution of inpainted images in an unsupervised setting, where only incomplete observations are available. The problem has been formulated using a Conditional Generative Adversarial Network setting. Training relies on the optimization of a criterion combining an adversarial loss allowing the recovery of sharp realistic signals, and a latent reconstruction loss allowing conditioning the reconstructed image to an incomplete observation. For the second one, we have proposed a GAN-based framework to complete par-

tially observed spatiotemporal data. Our model utilizes a generator to complete missing pixels in observation sequences with the help of two discriminators classifying real and generated observation sequences. We show that our model is able to complete spatiotemporal data without ground truth supervision when we have a stochastic model of the occlusion process.

6.2 Future Direction

Let us now briefly present extensions and future works that this thesis could lead to.

Hybrid Forecasting

In Chapter 4 and Chapter 3 we have proposed two forecasting systems. The first one uses prior knowledge from a PDE to model the evolution of a physical system, and the second use a continuous-time neural network in order to model the evolution of a dynamical system.

A natural perspective would be to consider a "hybridization" of the two approaches. Let us use the example of fluid dynamic. By considering jointly the evolution of the SST via an advection diffusion equation and the evolution of the velocity field via and dynamical systems, we can write the following system.

$$\begin{cases} \dfrac{\partial I}{\partial t} + (w.\nabla)I = D\nabla^2 I. \\ \dfrac{\partial w}{\partial t} = F(w, I, t). \end{cases} \tag{6.1}$$

Solving jointly this system could lead to strong forecasting performance by merging strong prior knowledge of a process and deep learning expressivity.

Data Assimilation and Inverse Problems

For future work we plan to apply our framework to different corruption processes, and evaluate our model's performance in real-world settings, specifically for retrieving scientific data from corrupted observation.

Our temporal model could be further tested using non-simulated stochastic operators with cloud masks extracted from satellite images or cloud data service.

It could be interesting to consider the task of unsupervised object removal. By mixing together segmentation network and inpainting network, we could remove unwanted objects from images without ever having access to images without those objects.

CHAPTER 7

ANNEX

Contents

7.1 Dataset of Section 4.3.1 119
7.2 Proof of Theorem Theorem 3.1 121
7.3 Proof of Theorem 4.1 124

7.1 Dataset of Section 4.3.1

7.1.0.1 The Shallow Water Equations

The shallow-water model can be written as:

$$\begin{aligned}
\frac{\partial u}{\partial t} &= +(f+\zeta).v - \partial_x(\frac{u^2+v^2}{2} + g^*.h) + \\
&\quad \frac{\tau_x}{\rho_0(H+h)} - \gamma.u + \nu\Delta u \\
\frac{\partial v}{\partial t} &= -(f+\zeta).u - \partial_y(\frac{u^2+v^2}{2} + g^*.h) + \\
&\quad \frac{\tau_y}{\rho_0(H+h)} - \gamma.v + \nu\Delta v \\
\frac{\partial h}{\partial t} &= -\partial_x(u(H+h)) - \partial_y(v(H+h))
\end{aligned} \tag{7.1}$$

where:

- u, v, h are state variables, standing for velocity and mixed layer depth anomaly)
- ζ is the vorticity.
- $g^* = 0.02$ is the reduced gravity
- $H = 500m$ is the mean mixed-layer depth.
- ρ_0 is the density of the water set to $1000mg/m^3$
- γ is the dissipation coefficient set to $2 \cdot 10^{-7}s^{-1}$

- ν is the diffusion coefficient set to $0.72m^2/s$
- τ_x is the zonal wind forcing defined in Eq. 7.1.0.1

The zonal wind forcing is defined as:

$$\tau_x(y) = \tau_0 \sin(2\pi(y - y_c)/L_y$$

where:

- τ_0 is the maximum intensity of the wind stress(in the standard case $0.15m.s^{-2}$).
- y is the latitude coordinate
- y_c is the center y coordinate of the domain
- L_y is the length of the domain ($L_y = 1600km$ in our case).

Here, the state is composed of the velocity vector and the mixed layer depth:

$$X = \begin{pmatrix} u \\ v \\ h \end{pmatrix} \text{ and } \mathcal{H}(X) = h$$

For our simulations, the spatial differential operators have been discretized using finite differences on a Arakawa C-grid.

7.1.0.2 The Navier-Stokes Equations

$$\begin{aligned} \frac{\partial u}{\partial t} + (u \cdot \nabla)u &= -\frac{\nabla p}{\rho} + g + \nu\nabla^2 u \\ \frac{\partial \rho}{\partial t} + (u \cdot \nabla)\rho &= 0 \\ \nabla \cdot u &= 0 \end{aligned} \tag{7.2}$$

where $\nabla\cdot$ is the divergence operator, u corresponds to the flow velocity vector, p to the pressure, and ρ to the density.

The Navier-Stokes equations are not of the form of equation 4.1 as we still have the pressure variable p as well as the null divergence constraint. However, the Helmholz-Leray decomposition result states that for any vector field a, there exists b and c such that :

$$a = \nabla b + c$$

and

$$\nabla \cdot c = 0$$

Moreover, this pair is unique up to an additive constant for b. Thus, we can define a linear operator $\mathbb{P}$ by :

$$\mathbb{P}(a) = c$$

This operator is a continuous linear projector which is the identity for divergence-free vector fields and vanishes for those deriving from a potential.

By taking a solution of equation 7.2 and applying $\mathbb{P}$ on the first equation, we have, as u is divergence free from the third equation and as g derives from a potential :

$$\frac{\partial u}{\partial t} = -\mathbb{P}[(u \cdot \nabla)u] + \nu \mathbb{P}(\nabla^2 u)$$

where permuting derivation and $\mathbb{P}$ is justified by the continuity of the operator [1].

Thus, if u is solution to equation 7.2, it is also a solution of :

$$\frac{\partial u}{\partial t} = -\mathbb{P}[(u \cdot \nabla)u] + \nu \mathbb{P}(\nabla^2 u)$$
$$\frac{\partial \rho}{\partial t} = -(u \cdot \nabla)\rho$$

which is of the form of equation 4.1.

Conversely, the solution of the above system is such that :

$$u_t = \int \frac{\partial u}{\partial t} = \int -\mathbb{P}[(u \cdot \nabla)u] + \nu \mathbb{P}(\nabla^2 u)$$

which gives, by exchanging $\mathbb{P}$ and the integral [2] :

$$u_t = \mathbb{P}\left[\int -(u \cdot \nabla)u + \nu \nabla^2 u\right]$$

so that u is automatically of null divergence by definition of $\mathbb{P}$. The two systems are thus equivalent.

In conclusion, we have:

$$X = \begin{pmatrix} u \\ \rho \end{pmatrix}, \text{ and } \mathcal{H}(X) = \rho$$

Moreover, u is generally a two or three-dimensional spatial field while ρ is a scalar field.

7.2 Proof of Theorem Theorem 3.1

In the following, bold $\mathbf{x}$ and $\mathbf{y}$ will denote vectors of $\mathbb{R}^2$, while x and y will correspond to the first and second components of $\mathbf{x}$, respectively. Analogously, u and v will correspond to the components of w. The 2D Fourier Transformation $\mathcal{F}$ of $f : \mathbb{R}^2 \to \mathbb{R}$ is defined as

1. One can use a finite difference approximation to show it for example.
2. To prove this, we can take a sum approximation to the integral and use again the linearity then the continuity of $\mathbb{P}$.

$$\begin{aligned}\mathcal{F}(f) &= \int_{\mathbb{R}^2} f(\mathbf{x}) e^{-i<\xi,\mathbf{x}>} d\mathbf{x} \\ &= \int_{\mathbb{R}} \int_{\mathbb{R}} f(x,y) e^{-ix\xi_1 - iy\xi_2} dxdy\end{aligned} \tag{7.3}$$

We apply the Fourier Transform $\mathcal{F}$ to both sides of Equation 3.3. As consequence of the linearity of the Fourier transform, we can calculate decompose the Fourier transform of the left hand side in the sum of the transforms of each term. We have three terms: $\frac{\partial I}{\partial t}$, $(w.\nabla)I$ and $-D\nabla^2 I$.

$$\begin{aligned}\mathcal{F}(\frac{\partial I}{\partial t}) &= \int_{\mathbb{R}^2} \frac{\partial I}{\partial t} e^{-i<\mathbf{x},\xi>} d\mathbf{x} \\ &= \int_{\mathbb{R}^2} \frac{\partial}{\partial t}(I e^{-i<\mathbf{x},\xi>}) d\mathbf{x} \\ &= \frac{\partial}{\partial t} \int_{\mathbb{R}^2} I e^{-i<\mathbf{x},\xi>} d\mathbf{x} \\ &= \frac{\partial \mathcal{F}(I)}{\partial t}\end{aligned} \tag{7.4}$$

$$\begin{aligned}\mathcal{F}((w.\nabla)I) &= \int_{\mathbb{R}^2} (w.\nabla) I e^{-i<\mathbf{x},\xi>} d\mathbf{x} \\ &= \int_{\mathbb{R}} \int_{\mathbb{R}} (u\frac{\partial I}{\partial x} + v\frac{\partial I}{\partial y}) e^{-ix\xi_1 - iy\xi_2} dxdy \\ &= u \int_{\mathbb{R}} e^{-iy\xi_2} \int_{\mathbb{R}} \frac{\partial I}{\partial x} e^{-ix\xi_1} dxdy + v \int_{\mathbb{R}} e^{-ix\xi_1} \int_{\mathbb{R}} \frac{\partial I}{\partial y} e^{-iy\xi_2} dydx \\ &= i\xi_1 u \int_{\mathbb{R}} e^{-iy\xi_2} \int_{\mathbb{R}} I e^{-ix\xi_1} dxdy + i\xi_2 v \int_{\mathbb{R}} e^{-ix\xi_1} \int_{\mathbb{R}} \frac{\partial I}{\partial y} e^{-iy\xi_2} dydx \\ &= i\xi_1 u \int_{\mathbb{R}} \int_{\mathbb{R}} I e^{-ix\xi_1 - iy\xi_2} dxdy + i\xi_2 v \int_{\mathbb{R}} \int_{\mathbb{R}} I e^{-ix\xi_1 - iy\xi_2} dxdy \\ &= (i\xi_1 u + i\xi_2 v) \int_{\mathbb{R}} \int_{\mathbb{R}} I e^{-ix\xi_1 - iy\xi_2} dxdy \\ &= i < \xi, w > \mathcal{F}(I)\end{aligned} \tag{7.5}$$

$$\begin{aligned}
\mathcal{F}(-D\nabla^2 I) &= -\int_{\mathbb{R}^2} D\nabla^2 I e^{-i<\mathbf{x},\xi>} d\mathbf{x} \\
&= -\int_{\mathbb{R}}\int_{\mathbb{R}} D(\frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}) e^{-ix\xi_1 - iy\xi_2} dxdy \\
&= -D\int_{\mathbb{R}} e^{-iy\xi_2}\int_{\mathbb{R}} \frac{\partial^2 I}{\partial x^2} e^{-ix\xi_1} dxdy - D\int_{\mathbb{R}} e^{-ix\xi_1}\int_{\mathbb{R}} \frac{\partial^2 I}{\partial y^2} e^{-iy\xi_2} dydx \\
&= -(i\xi_1)^2 D\int_{\mathbb{R}} e^{-iy\xi_2}\int_{\mathbb{R}} I e^{-ix\xi_1} dxdy - (i\xi_2)^2 D\int_{\mathbb{R}} e^{-ix\xi_1}\int_{\mathbb{R}} I e^{-iy\xi_2} dydx \\
&= D\xi_1^2\int_{\mathbb{R}} e^{-iy\xi_2}\int_{\mathbb{R}} I e^{-ix\xi_1} dxdy + D\xi_2^2\int_{\mathbb{R}} e^{-ix\xi_1}\int_{\mathbb{R}} I e^{-iy\xi_2} dydx \\
&= D\xi_1^2\int_{\mathbb{R}}\int_{\mathbb{R}} I e^{-ix\xi_1 - iy\xi_2} dxdy + D\xi_2^2\int_{\mathbb{R}}\int_{\mathbb{R}} I e^{-ix\xi_1 - iy\xi_2} dxdy \\
&= D\left\|\xi\right\|^2 \mathcal{F}(I)
\end{aligned} \tag{7.6}$$

Regrouping all three previously calculated terms, we obtain

$$\frac{\partial \mathcal{F}(I)}{\partial t} + (i < \xi, w > + D\left\|\xi\right\|^2)\mathcal{F}(I) = 0 \tag{7.7}$$

This is a first order ordinary differential equation of the form $f'(t) + af(t) = 0$, which admits a known solution $f(t) = f(0)e^{-at}$. Thus, the solution of Equation 7.7 is

$$\begin{aligned}
\mathcal{F}(I) &= \mathcal{F}(I)_0\, e^{-(i<\xi,w> + D\left\|\xi\right\|^2)t} \\
&= \mathcal{F}(I)_0\, e^{-i<\xi,w>t} e^{-Dt\left\|\xi\right\|^2}
\end{aligned} \tag{7.8}$$

where $\mathcal{F}(I)_0$ denotes the initial condition of the advection diffusion equation in the frequency domain. In order to obtain a solution of Equation 3.3 in the spatial domain, we calculate the inverse Fourier Transform $\mathcal{F}^{-1}$ of Equation 7.8. The multiplication of two functions in the frequency domain is equivalent to their convolution in the spatial domain, *i.e.* $\mathcal{F}(f * g) = \mathcal{F}(f)\mathcal{F}(g)$. Hence, the inverse of both terms $\mathcal{F}(I)_0\, e^{-i<\xi,w>t}$ and $e^{-Dt\left\|\xi\right\|^2}$ can be calculated separately:

Multiplication by a complex exponential in the frequency domain is equivalent to a shift in the spatial domain : $e^{-i<\xi,w>}\mathcal{F}(f(\mathbf{x})) = \mathcal{F}(f(\mathbf{x} - w))$, for $v \in \mathbb{R}^2$. Thus, for the first term,

$$\mathcal{F}^{-1}(\mathcal{F}(I)_0\, e^{-(i<\xi,w>)t}) = I_0(\mathbf{x} - w) \tag{7.9}$$

For the second term, we use the fact that the Fourier Transform of a Gaussian function also is a Gaussian function, *i.e.* $\mathcal{F}(\frac{1}{2\pi\sigma^2} e^{-\frac{1}{2\sigma^2}\left\|\mathbf{x}\right\|^2}) = e^{-\frac{1}{2}\sigma^2\left\|\xi\right\|^2}$. Identifying σ^2 with $2Dt$, we have:

$$\mathcal{F}^{-1}(e^{-Dt\|\xi\|^2}) = \frac{1}{4\pi Dt} e^{-\frac{1}{4Dt}\|\mathbf{x}\|^2} \tag{7.10}$$

As has been stated above, the solution is a convolution of both previously calculated terms:

$$\begin{aligned} I(\mathbf{x}, t) &= \int_{\mathbb{R}^2} \frac{1}{4\pi Dt} e^{-\frac{1}{4Dt}\|\mathbf{y}\|^2} I_0(\mathbf{x} - w - \mathbf{y}) dy \\ &= \int_{\mathbb{R}^2} \frac{1}{4\pi Dt} e^{-\frac{1}{4Dt}\|\mathbf{x}-w-\mathbf{y}\|^2} I_0(\mathbf{y}) d\mathbf{y} \end{aligned} \tag{7.11}$$

which concludes the proof. □

7.3 Proof of Theorem 4.1

In order to use gradient descent, we must first calculate the gradient of the cost functional under the constraints, *i.e.* the differential of $\theta \to \mathbb{E}_Y \mathcal{J}(Y, \mathcal{H}(X^\theta))$. However, this implies calculating $\frac{\partial X^\theta}{\partial \theta}$, which is often very computationally demanding, as it implies solving $\dim(\theta)$ forward equations. The adjoint state method avoids those costly computations by considering the Lagrangian formulation of the constrained optimization problem introduced in equation 4.5, the Lagrangian being defined as:

$$\begin{aligned} \mathcal{L}(X, \lambda, \mu, \theta) = \mathcal{J}(X) + \int_0^T \left\langle \lambda_t, \frac{dX_t}{dt} - F_\theta(X_t) \right\rangle \mathrm{dt} \\ + \langle \mu, X_0 - g_\theta \rangle \end{aligned} \tag{7.12}$$

here, the scalar product $\langle \cdot, \cdot \rangle$ is the scalar product associated to the L^2 space over Ω.

As, for any θ, X^θ satisfies the constraints by definition, we can now write:

$$\forall \theta, \lambda, \mu, \ \mathcal{L}(X^\theta, \lambda, \mu, \theta) = \mathcal{J}(Y, \mathcal{H}(X^\theta))$$

which gives :

$$\forall \lambda, \mu, \ \frac{\partial}{\partial \theta} \mathcal{L}(X^\theta, \lambda, \mu, \theta) = \frac{\partial}{\partial \theta} \mathcal{J}(X^\theta)$$

Now we have to calculate the differential of $\mathcal{L}$ *w.r.t.* θ and use it to have the gradient of $\mathcal{J}$. We start by differentiating $\mathcal{L}$. In what follows, all considered functions are supposed to be twice continuously differentiable in all variables and we will use the notation $\partial_u F(u_0)$ to designate the differential of F with respect to u *i.e.* the unique linear operator such that:

$$F(u_0 + \delta u) = F(u_0) + \partial_u F(u_0)\delta u + o(\delta u)$$

By hypothesis, we consider this operator to be always continuous in our case.

Straightforward calculus gives us:

$$\frac{\partial \mathcal{J}(X_t^\theta)}{\partial \theta} = \int_0^T 2 \left\langle \partial_X \mathcal{H}(X_t^\theta) \cdot \partial_\theta X_t^\theta, \mathcal{H}(X_t^\theta) - Y_t \right\rangle \mathrm{d}t$$

Let us fix θ and a variation $\delta\theta$. Then, we have, by definition:

$$X^{\theta+\delta\theta} = X_t^\theta + \partial_\theta X_t^\theta \cdot \delta\theta + o(\delta\theta)$$

and, for any X and any δX:

$$F_\theta(X + \delta X) = F_\theta(X) + \partial_X F_\theta(X) \cdot \delta X + o(\delta X)$$

and:

$$F_{\theta+\delta\theta}(X) = F_\theta(X) + \partial_\theta F_\theta(X) \cdot \delta\theta + o(\delta\theta)$$

so that:

$$F_{\theta+\delta\theta}(X_t^{\theta+\delta\theta}) = F_\theta(X_t^{\theta+\delta\theta}) + \partial_\theta F_\theta(X_t^{\theta+\delta\theta}) \cdot \delta\theta + o(\delta\theta)$$

Then, because F is twice continuously differentiable:

$$\begin{aligned}
\partial_\theta F_\theta(X_t^{\theta+\delta\theta}) &= \partial_\theta F_\theta \left(X_t^\theta + \partial_\theta X_t^\theta \cdot \delta\theta + o(\delta\theta) \right) \\
&= \partial_\theta F_\theta(X_t^\theta) + \partial_X \partial_\theta F_\theta(X_t^\theta) \cdot \partial_\theta X_t^\theta \cdot \delta\theta \\
&\quad + o(\delta\theta)
\end{aligned}$$

and:

$$\begin{aligned}
F_\theta(X_t^{\theta+\delta\theta}) &= F_\theta \left(X_t^\theta + \partial_\theta X_t^\theta \cdot \delta\theta + o(\delta\theta) \right) \\
&= F_\theta(X_t^\theta) + \partial_X F_\theta(X_t^\theta) \cdot \partial_\theta X_t^\theta \cdot \delta\theta + o(\delta\theta)
\end{aligned}$$

Moreover, as all differential operators below are continuous by hypothesis, we have that:

$$\| (\partial_X \partial_\theta F_\theta(X_t^\theta) \cdot \partial_\theta X_t^\theta \cdot \delta\theta) \cdot \delta\theta \| \leq \| \partial_X \partial_\theta F_\theta(X_t^\theta) \| \, \| \partial_\theta X_t^\theta \| \, \| \delta\theta \|^2$$

so that:

$$\begin{aligned}
&F_{\theta+\delta\theta}(X_t^{\theta+\delta\theta}) \\
&\quad = F_\theta(X_t^\theta) + \left(\partial_X F_\theta(X_t^\theta) \cdot \partial_\theta X_t^\theta + \partial_\theta F_\theta(X_t^\theta) \right) \cdot \delta\theta + o(\delta\theta)
\end{aligned}$$

We now have all elements to conclude calculating the derivative of $\mathcal{L}$, with some more easy calculus:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \theta} = \int_0^T \Big(& 2 \left\langle \partial_X \mathcal{H}(X_t^\theta) \cdot \partial_\theta X_t^\theta, \mathcal{H}(X_t^\theta) - Y_t \right\rangle + \\
& \left\langle \lambda_t, \partial_\theta \partial_t X_t^\theta - \partial_X F_\theta(X_t^\theta) \cdot \partial_\theta X_t^\theta - \partial_\theta F_\theta(X_t^\theta) \right\rangle \Big) \mathrm{d}t \\
& + \left\langle \mu, \partial_\theta X_0^\theta - \partial_\theta g_\theta \right\rangle
\end{aligned}$$

By the Schwarz theorem, as X is twice continuously differentiable, we have that $\partial_\theta \partial_t X_t^\theta = \partial_t \partial_\theta X_t^\theta$. Integrating by parts, we get:

$$\int_0^T \left\langle \lambda_t, \partial_\theta \partial_t X_t^\theta \right\rangle \mathrm{d}t = \left\langle \lambda_T, \partial_\theta X_T^\theta \right\rangle - \left\langle \lambda_0, \partial_\theta X_0^\theta \right\rangle - \int_0^T \left\langle \partial_t \lambda_t, \partial_\theta X_t^\theta \right\rangle \mathrm{d}t$$

Putting all this together and arranging it, we get:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \theta} = &\int_0^T \Big\langle \partial_\theta X_t^\theta, 2\partial_X \mathcal{H}(X_t^\theta)^\star \left(\mathcal{H}(X_t^\theta) - Y_t\right) \\ &-\partial_t \lambda_t - \partial_X F_\theta(X_t^\theta)^\star \lambda_t \Big\rangle \mathrm{d}t \\ &- \int_0^T \left\langle \lambda_t, \partial_\theta F_\theta(X_t^\theta) \right\rangle \mathrm{d}t + \left\langle \lambda_T, \partial_\theta X_T^\theta \right\rangle + \left\langle \mu - \lambda_0, \partial_\theta X_0^\theta \right\rangle \\ &- \langle \mu, \partial_\theta g_\theta \rangle\end{aligned}$$

We can now define:

$$A_t = -(\partial_X F_\theta(X_t^\theta))^\star$$

and

$$B_t = 2(\partial_X \mathcal{H}(X_t^\theta))^\star (\mathcal{H}(X_t^\theta) - Y_t)$$

and, recalling that λ can be freely chosen, impose that λ is solution of:

$$\partial_t \lambda_t = A_t \lambda_t + B_t$$

with final condition $\lambda_T = 0$. We also choose $\mu = \lambda_0$ so that, finally, we have:

$$\frac{\partial \mathcal{L}}{\partial \theta} = -\int_0^T \left\langle \lambda_t, \partial_\theta F_\theta(X_t^\theta) \right\rangle \mathrm{d}t - \langle \lambda_0, \partial_\theta g_\theta \rangle$$

which concludes the proof. □

REFERENCES

Aach, John and George M. Church (June 2001). "Aligning gene expression time series with time warping algorithms". In: *Bioinformatics* 17.6, pp. 495–508. URL: https://academic.oup.com/bioinformatics/article/17/6/495/272361 (cit. on p. 8).

Almahairi, Amjad, Sai Rajeswar, Alessandro Sordoni, Philip Bachman, and Aaron Courville (2018). "Augmented CycleGAN: Learning Many-to-Many Mappings from Unpaired Data". In: *arXiv preprint arXiv:1802.10151* (cit. on pp. 21, 22, 79).

Alvarez, Mauricio A., David Luengo, and Neil D. Lawrence (Nov. 2013). "Linear Latent Force Models Using Gaussian Processes". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 35.11, pp. 2693–2705 (cit. on p. 18).

Alvera Azcarate, Aïda, Alexander Barth, Michel Rixen, and Jean-Marie Beckers (2005). "Reconstruction of incomplete oceanographic data sets using empirical orthogonal functions: application to the Adriatic Sea surface temperature". In: *Ocean Modelling* 9.4. URL: https://orbi.uliege.be/handle/2268/4296 (cit. on pp. 93, 102, 103).

Asim, Muhammad, Fahad Shamshad, and Ali Ahmed (2018). "Solving Bilinear Inverse Problems using Deep Generative Priors". In: *CoRR* abs/1802.04073. URL: http://arxiv.org/abs/1802.04073 (cit. on pp. 23, 67).

Ayed, Ibrahim, Emmanuel de Bézenac, Arthur Pajot, Julien Brajard, and Patrick Gallinari (Feb. 2019). "Learning Dynamical Systems from Partial Observations". In: *arXiv:1902.11136 [physics]*. URL: http://arxiv.org/abs/1902.11136 (cit. on pp. 4, 48).

Ballester, Coloma, Marcelo Bertalmio, Vicent Caselles, Guillermo Sapiro, and Joan Verdera (2000). "Filling-in by joint interpolation of vector fields and gray levels". In: (cit. on p. 24).

Barnes, Connelly, Eli Shechtman, Adam Finkelstein, and Dan B Goldman (2009). "PatchMatch: A randomized correspondence algorithm for structural image editing". In: *ACM Transactions on Graphics (ToG)*. Vol. 28, p. 24 (cit. on p. 24).

Béréziat, Dominique and Isabelle Herlin (2015). "Coupling Dynamic Equations and Satellite Images for Modelling Ocean Surface Circulation". In: *Computer Vision, Imaging and Computer Graphics - Theory and Applications: International Joint Conference, VISIGRAPP 2014, Lisbon, Portugal, January 5-8, 2014, Revised Selected Papers*. Ed. by Sebastiano Battiato, Sabine Coquillart, Julien Pettré, Robert S. Laramee, Andreas Kerren, and José Braz. Cham: Springer International Publishing, pp. 191–205 (cit. on pp. 38, 40).

Bernstein, R. L. (1982). "Sea surface temperature estimation using the NOAA 6 satellite advanced very high resolution radiometer". In: *Journal of Geophysical*

Research: Oceans 87 (C12), pp. 9455–9465. URL: http://dx.doi.org/10.1029/JC087iC12p09455 (cit. on pp. 29, 35).

Bertalmio, Marcelo, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester (2000). "Image inpainting". In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pp. 417–424 (cit. on p. 24).

Bezenac, Emmanuel de, Arthur Pajot, and Patrick Gallinari (Feb. 2018). "Deep Learning for Physical Processes: Incorporating Prior Scientific Knowledge". In: URL: https://openreview.net/forum?id=By4HsfWAZ (cit. on pp. 3, 28).

Bocquet, M (2012). "Parameter-field estimation for atmospheric dispersion: Application to the Chernobyl accident using 4D-Var". In: *Quarterly Journal of the Royal Meteorological Society* 138.664, pp. 664–681. URL: https://rmets.onlinelibrary.wiley.com/doi/pdf/10.1002/qj.961 (cit. on p. 18).

Bora, Ashish, Ajil Jalal, Eric Price, and Alexandros G. Dimakis (Mar. 2017). "Compressed Sensing using Generative Models". In: *arXiv:1703.03208 [cs, math, stat]*. URL: http://arxiv.org/abs/1703.03208 (cit. on pp. 23, 67, 90).

Bora, Ashish, Eric Price, and Alexandros G. Dimakis (2018). "AmbientGAN: Generative models from lossy measurements". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=Hy7fDog0b (cit. on pp. 24, 73, 83, 85, 90).

Boyat, Ajay Kumar and Brijendra Kumar Joshi (2015). "A Review Paper: Noise Models in Digital Image Processing". In: *CoRR* abs/1505.03489 (cit. on p. 70).

Candès, Emmanuel J., Justin K. Romberg, and Terence Tao (2005). "Stable signal recovery from incomplete and inaccurate measurements". In: *Communications on Pure and Applied Mathematics* 59.8, pp. 1207–1223. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpa.20124 (cit. on p. 67).

Carrassi, Alberto, Marc Bocquet, Laurent Bertino, and Geir Evensen (2018). "Data assimilation in the geosciences: An overview of methods, issues, and perspectives". In: *Wiley Interdisciplinary Reviews: Climate Change* 9.5, e535. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/wcc.535 (cit. on p. 52).

Chambolle, Antonin (Jan. 2004). "An Algorithm for Total Variation Minimization and Applications". In: *Journal of Mathematical Imaging and Vision* 20.1, pp. 89–97. URL: https://doi.org/10.1023/B:JMIV.0000011325.36760.1e (cit. on p. 92).

Chen, Tian Qi, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud (June 2018). "Neural Ordinary Differential Equations". In: *arXiv:1806.07366 [cs, stat]*. URL: http://arxiv.org/abs/1806.07366 (cit. on pp. 14, 15, 52, 59).

Cho, Kyunghyun, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio (Oct. 2014). "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". In: pp. 1724–1734. URL: https://aclweb.org/anthology/papers/D/D14/D14-1179/ (cit. on p. 9).

Chu, Casey, Andrey Zhmoginov, and Mark Sandler (2017). "Cyclegan, a master of steganography". In: *arXiv preprint arXiv:1712.02950* (cit. on p. 79).

Connor, Jerome T., R. Douglas Martin, and L. E. Atlas (1994). "Recurrent neural networks and robust time series prediction". In: *Ieee Transactions on Neural Networks* 5, pp. 240–254 (cit. on p. 9).

Cressie, Noel and Christopher K. Wikle (Nov. 2015). *Statistics for Spatio-Temporal Data*. John Wiley & Sons (cit. on pp. 9, 13).

Crutchfield, James P. and Bruce S. Mcnamara (1987). "Equations of motion from a data series". In: *Complex Systems*, p. 452 (cit. on p. 18).

Damelin, SB and NS Hoang (2018). "On surface completion and image inpainting by biharmonic functions: Numerical aspects". In: *International Journal of Mathematics and Mathematical Sciences* 2018 (cit. on p. 91).

Demir, Ugur and Gozde Unal (Mar. 2018). "Patch-Based Image Inpainting with Generative Adversarial Networks". In: *arXiv:1803.07422 [cs]*. URL: http://arxiv.org/abs/1803.07422 (cit. on p. 70).

Dey, Rahul and Fathi M. Salem (Jan. 2017). "Gate-Variants of Gated Recurrent Unit (GRU) Neural Networks". In: *arXiv:1701.05923 [cs, stat]*. URL: http://arxiv.org/abs/1701.05923 (cit. on p. 10).

Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio (2016). "Density estimation using Real NVP". In: *CoRR* abs/1605.08803. URL: http://arxiv.org/abs/1605.08803 (cit. on p. 67).

Dosovitskiy, Alexey, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox (Dec. 2015). "FlowNet: Learning Optical Flow with Convolutional Networks". In: IEEE, pp. 2758–2766. URL: http://ieeexplore.ieee.org/document/7410673/ (cit. on p. 11).

Ebert, Frederik, Chelsea Finn, Alex X. Lee, and Sergey Levine (Oct. 14, 2017). "Self-Supervised Visual Planning with Temporal Skip Connections". In: *arXiv:1710.05268 [cs]*. arXiv: 1710.05268. URL: http://arxiv.org/abs/1710.05268 (cit. on p. 86).

Fablet, Ronan, Said Ouala, and Cédric Herzet (2017). "Bilinear residual Neural Network for the identification and forecasting of dynamical systems". In: *CoRR* abs/1712.07003. URL: http://arxiv.org/abs/1712.07003 (cit. on p. 18).

Finn, Chelsea, Ian J. Goodfellow, and Sergey Levine (2016). "Unsupervised Learning for Physical Interaction through Video Prediction". In: *CoRR* abs/1605.07157. URL: http://arxiv.org/abs/1605.07157 (cit. on pp. 12, 34).

Goodfellow, Ian J., Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2014). "Generative Adversarial Nets". In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'14. MIT Press,

pp. 2672–2680. URL: http://dl.acm.org/citation.cfm?id=2969033.2969125 (cit. on pp. 20, 67).

Grathwohl, Will, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud (Oct. 2018). "FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models". In: *arXiv:1810.01367 [cs, stat]*. URL: http://arxiv.org/abs/1810.01367 (cit. on p. 14).

Graves, Alex, Abdel-rahman Mohamed, and Geoffrey Hinton (May 2013). "Speech recognition with deep recurrent neural networks". In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. Vancouver, BC, Canada: IEEE, pp. 6645–6649. URL: http://ieeexplore.ieee.org/document/6638947/ (cit. on p. 9).

Hays, James and Alexei A Efros (2007). "Scene Completion Using Millions of Photographs". In: *ACM Transactions on Graphics (SIGGRAPH 2007)* 26.3 (cit. on p. 24).

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (Dec. 2015). "Deep Residual Learning for Image Recognition". In: *arXiv:1512.03385 [cs]*. URL: http://arxiv.org/abs/1512.03385 (cit. on pp. 4, 14, 37, 55).

Heusel, Martin, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Günter Klambauer, and Sepp Hochreiter (2017). "GANs Trained by a Two Time-Scale Update Rule Converge to a Nash Equilibrium". In: *CoRR* abs/1706.08500. URL: http://arxiv.org/abs/1706.08500 (cit. on p. 89).

Hochreiter, Sepp and Jürgen Schmidhuber (Nov. 1997). "Long Short-Term Memory". In: *Neural Comput.* 9.8, pp. 1735–1780. URL: http://dx.doi.org/10.1162/neco.1997.9.8.1735 (cit. on p. 10).

Iizuka, Satoshi, Edgar Simo-Serra, and Hiroshi Ishikawa (July 2017). "Globally and Locally Consistent Image Completion". In: *ACM Trans. Graph.* 36.4, 107:1–107:14. URL: http://doi.acm.org/10.1145/3072959.3073659 (cit. on p. 24).

Ilg, Eddy, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox (2016). "FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks". In: *CoRR* abs/1612.01925. URL: http://arxiv.org/abs/1612.01925 (cit. on p. 11).

Isola, Phillip, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros (2016). "Image-to-Image Translation with Conditional Adversarial Networks". In: *CoRR* abs/1611.07004. URL: http://arxiv.org/abs/1611.07004 (cit. on pp. 21, 24, 85, 90, 92, 93).

Jaderberg, Max, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu (2015). "Spatial Transformer Networks". In: *CoRR* abs/1506.02025. URL: http://arxiv.org/abs/1506.02025 (cit. on pp. 12, 34).

Kalchbrenner, Nal, Aäron van den Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu (2016). "Video Pixel

Networks". In: *CoRR* abs/1610.00527. URL: http://arxiv.org/abs/1610.00527 (cit. on p. 12).

Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (cit. on p. 85).

Kingma, Diederik P and Max Welling (2013). "Auto-Encoding Variational Bayes". In: *arXiv preprint arXiv:1312.6114* (cit. on p. 67).

Ladický, L'ubor, SoHyeon Jeong, Barbara Solenthaler, Marc Pollefeys, and Markus Gross (Oct. 2015). "Data-driven Fluid Simulations Using Regression Forests". In: *ACM Trans. Graph.* 34.6, 199:1–199:9. URL: http://doi.acm.org/10.1145/2816795.2818129 (cit. on p. 18).

"Large Scale GAN Training for High Fidelity Natural Image Synthesis" (n.d.). In: (). URL: http://arxiv.org/abs/1809.11096 (cit. on p. 21).

LeCun, Yann, D Touresky, G Hinton, and T Sejnowski (1988). "A theoretical framework for back-propagation". In: *Proceedings of the 1988 connectionist models summer school*. Vol. 1. CMU, Pittsburgh, Pa: Morgan Kaufmann, pp. 21–28 (cit. on p. 53).

Ledig, Christian, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew P. Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi (2016). "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network". In: *CoRR* abs/1609.04802. URL: http://arxiv.org/abs/1609.04802 (cit. on p. 14).

Lehtinen, Jaakko, Jacob Munkberg, Jon Hasselgren, Samuli Laine, Tero Karras, Miika Aittala, and Timo Aila (Mar. 2018). "Noise2Noise: Learning Image Restoration without Clean Data". In: *arXiv:1803.04189 [cs, stat]*. URL: http://arxiv.org/abs/1803.04189 (cit. on p. 23).

Li, Steven Cheng-Xian, Bo Jiang, and Benjamin Marlin (Sept. 2018). "MisGAN: Learning from Incomplete Data with Generative Adversarial Networks". In: URL: https://openreview.net/forum?id=S1lDV3RcKm (cit. on pp. 24, 93).

Ling, Julia, Andrew Kurzawski, and Jeremy Templeton (Nov. 2016). "Reynolds averaged turbulence modelling using deep neural networks with embedded invariance". In: *Journal of Fluid Mechanics* 807, pp. 155–166 (cit. on p. 34).

Liu, Guilin, Fitsum A. Reda, Kevin J. Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro (Apr. 2018). "Image Inpainting for Irregular Holes Using Partial Convolutions". In: *arXiv:1804.07723 [cs]*. URL: http://arxiv.org/abs/1804.07723 (cit. on p. 24).

Liu, Ziwei, Ping Luo, Xiaogang Wang, and Xiaoou Tang (2015). "Deep Learning Face Attributes in the Wild". In: *Proceedings of International Conference on Computer Vision (ICCV)* (cit. on pp. 67, 85).

Long, Zichao, Yiping Lu, Xianzhong Ma, and Bin Dong (2018). "PDE-Net: Learning PDEs from Data". In: *ICML*, pp. 3214–3222 (cit. on p. 18).

Madec, G. (2008). *NEMO ocean engine*. Note du Pôle de modélisation, Institut Pierre-Simon Laplace (IPSL), France, No 27, ISSN No 1288-1619 (cit. on p. 35).

Mao, Xudong, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley (2017). "Least squares generative adversarial networks". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2794–2802 (cit. on p. 20).

Mardani, Morteza, Enhao Gong, Joseph Y. Cheng, Shreyas Vasanawala, Greg Zaharchuk, Marcus T. Alley, Neil Thakur, Song Han, William J. Dally, John M. Pauly, and Lei Xing (2017). "Deep Generative Adversarial Networks for Compressed Sensing Automates MRI". In: *CoRR* abs/1706.00051. URL: http://arxiv.org/abs/1706.00051 (cit. on p. 23).

Marin, Javier, Aritro Biswas, Ferda Ofli, Nicholas Hynes, Amaia Salvador, Yusuf Aytar, Ingmar Weber, and Antonio Torralba (2018). "Recipe1M: A Dataset for Learning Cross-Modal Embeddings for Cooking Recipes and Food Images". In: *arXiv preprint arXiv:1810.06553* (cit. on pp. 67, 86).

Mathieu, Michaël, Camille Couprie, and Yann LeCun (2015). "Deep multi-scale video prediction beyond mean square error". In: *CoRR* abs/1511.05440. URL: http://arxiv.org/abs/1511.05440 (cit. on pp. 12, 38).

Miyato, Takeru, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida (2018). "Spectral Normalization for Generative Adversarial Networks". In: *CoRR* abs/1802.05957. URL: http://arxiv.org/abs/1802.05957 (cit. on p. 85).

Mota, João FC, Nikos Deligiannis, and Miguel RD Rodrigues (2017). "Compressed sensing with prior information: Strategies, geometry, and bounds". In: *IEEE Transactions on Information Theory* 63.7, pp. 4472–4496 (cit. on p. 67).

Newson, Alasdair, Andrés Almansa, Matthieu Fradet, Yann Gousseau, and Patrick Pérez (Jan. 2014). "Video Inpainting of Complex Scenes". In: *SIAM Journal on Imaging Sciences* 7.4, pp. 1993–2019. URL: http://epubs.siam.org/doi/10.1137/140954933 (cit. on pp. 93, 102–104).

Pajot, Arthur, Emmanuel de Bezenac, and Patrick Gallinari (Sept. 2018). "Unsupervised Adversarial Image Reconstruction". In: URL: https://openreview.net/forum?id=BJg4Z3RqF7 (cit. on pp. 5, 66, 83, 105).

Paszke, Adam, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer (2017). "Automatic differentiation in PyTorch". In: (cit. on pp. 52, 55).

Pathak, Deepak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros (2016). "Context encoders: Feature learning by inpainting". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2536–2544 (cit. on p. 24).

Patraucean, Viorica, Ankur Handa, and Roberto Cipolla (2015). "Spatio-temporal video autoencoder with differentiable memory". In: *CoRR* abs/1511.06309. URL: http://arxiv.org/abs/1511.06309 (cit. on pp. 12, 34).

Pressel, Kyle G., Colleen M. Kaul, Tapio Schneider, Zhihong Tan, and Siddhartha Mishra (2015). "Large-eddy simulation in an anelastic framework with closed water and entropy balances". In: *Journal of Advances in Modeling Earth Systems* 7.3, pp. 1425–1456. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2015MS000496 (cit. on p. 88).

Raissi, Maziar (Apr. 2018). "Forward-Backward Stochastic Neural Networks: Deep Learning of High-dimensional Partial Differential Equations". In: *arXiv:1804.07010 [cs, math, stat]*. URL: http://arxiv.org/abs/1804.07010 (cit. on p. 18).

Raissi, Maziar, Paris Perdikaris, and George Em Karniadakis (2017). "Machine learning of linear differential equations using Gaussian processes". In: *Journal of Computational Physics* 348, pp. 683–693 (cit. on p. 18).

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *CoRR* abs/1505.04597. URL: http://arxiv.org/abs/1505.04597 (cit. on p. 61).

Rössler, Andreas, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Niet sner (2019). "FaceForensics++: Learning to Detect Manipulated Facial Images". In: *CoRR* abs/1901.08971 (cit. on p. 86).

Rudy, Samuel H., Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz (Apr. 2017). "Data-driven discovery of partial differential equations". In: *Science Advances* 3.4, e1602614 (cit. on p. 18).

Schuldt, C., I. Laptev, and B. Caputo (2004). "Recognizing human actions: a local SVM approach". In: *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.* Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004. Cambridge, UK: IEEE, 32–36 Vol.3. URL: http://ieeexplore.ieee.org/document/1334462/ (cit. on p. 86).

Shi, Wenzhe, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang (2016). "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1874–1883 (cit. on p. 23).

Shi, Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo (2015). "Convolutional LSTM network: A machine learning approach for precipitation nowcasting". In: *Advances in Neural Information Processing Systems 28*, pp. 802–810 (cit. on pp. 12, 38, 40).

Simakov, Denis, Yaron Caspi, Eli Shechtman, and Michal Irani (2008). "Summarizing visual data using bidirectional similarity". In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8 (cit. on p. 24).

Sirkes, Ziv and Eli Tziperman (1997). "Finite difference of adjoint or adjoint of finite difference?" In: *Monthly weather review* 125.12, pp. 3373–3378 (cit. on p. 52).

Sønderby, Casper Kaae, Jose Caballero, Lucas Theis, Wenzhe Shi, and Ferenc Huszár (Oct. 2016). "Amortised MAP Inference for Image Super-resolution". In: *arXiv:1610.04490 [cs, stat]*. URL: http://arxiv.org/abs/1610.04490 (cit. on p. 23).

Song, Yuhang, Chao Yang, Zhe Lin, Hao Li, Qin Huang, and C.-C. Jay Kuo (2017). "Image Inpainting using Multi-Scale Feature Image Translation". In: *CoRR* abs/1711.08590. URL: http://arxiv.org/abs/1711.08590 (cit. on p. 24).

Stuart, A. M. (2010). "Inverse problems: A Bayesian perspective". In: *Acta Numerica* 19, pp. 451–559 (cit. on p. 67).

Sun, Deqing, Stefan Roth, J. P. Lewis, and Michael J. Black (2008). "Learning Optical Flow". In: *Computer Vision – ECCV 2008: 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part III*. Ed. by David Forsyth, Philip Torr, and Andrew Zisserman. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 83–97. URL: https://doi.org/10.1007/978-3-540-88690-7_7 (cit. on p. 11).

Sutskever, Ilya, James Martens, and Geoffrey Hinton (n.d.). "Generating Text with Recurrent Neural Networks". In: (), p. 8 (cit. on p. 9).

Thissen, U, R van Brakel, A. P de Weijer, W. J Melssen, and L. M. C Buydens (Nov. 2003). "Using support vector machines for time series prediction". In: *Chemometrics and Intelligent Laboratory Systems* 69.1, pp. 35–49. URL: http://www.sciencedirect.com/science/article/pii/S0169743903001114 (cit. on p. 9).

Tompson, Jonathan, Kristofer Schlachter, Pablo Sprechmann, and Ken Perlin (Aug. 2017). "Accelerating Eulerian Fluid Simulation With Convolutional Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, pp. 3424–3433. URL: http://proceedings.mlr.press/v70/tompson17a.html (cit. on p. 18).

"Towards Accurate Generative Models of Video" (n.d.). "Towards Accurate Generative Models of Video: A New Metric & Challenges". In: (). URL: http://arxiv.org/abs/1812.01717 (cit. on p. 89).

Trémolet, Yannick (2006). "Accounting for an imperfect model in 4D-Var". In: *Quarterly Journal of the Royal Meteorological Society* 132.621, pp. 2483–2504 (cit. on p. 38).

Tripathi, Subarna, Zachary C. Lipton, and Truong Q. Nguyen (2018). "Correction by Projection: Denoising Images with Generative Adversarial Networks". In: *CoRR* abs/1803.04477. URL: http://arxiv.org/abs/1803.04477 (cit. on pp. 23, 67).

Vallis, Geoffrey K. (2017). *Atmospheric and Oceanic Fluid Dynamics: Fundamentals and Large-Scale Circulation*. 2nd ed. Cambridge University Press (cit. on p. 38).

Van Amersfoort, Joost, Anitha Kannan, Marc'Aurelio Ranzato, Arthur Szlam, Du Tran, and Soumith Chintala (2017). "Transformation-Based Models of Video Sequences". In: *CoRR abs/1701.08435 (2017)*, pp. 1–11. URL: http://arxiv.org/abs/1701.08435 (cit. on p. 12).

Van Veen, David, Ajil Jalal, Eric Price, Sriram Vishwanath, and Alexandros G. Dimakis (June 2018). "Compressed Sensing with Deep Image Prior and Learned Regularization". In: *arXiv:1806.06438 [cs, math, stat]*. URL: http://arxiv.org/abs/1806.06438 (cit. on pp. 23, 90, 91).

Wang, Ting-Chun, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro (Aug. 20, 2018). "Video-to-Video Synthesis". In: *arXiv:1808.06601 [cs]*. arXiv: 1808.06601. URL: http://arxiv.org/abs/1808.06601 (cit. on p. 94).

Wang, Yunbo, Zhifeng Gao, Mingsheng Long, Jianmin Wang, and Philip S. Yu (Apr. 2018). "PredRNN++: Towards A Resolution of the Deep-in-Time Dilemma in Spatiotemporal Predictive Learning". In: *arXiv:1804.06300 [cs, stat]*. URL: http://arxiv.org/abs/1804.06300 (cit. on pp. 12, 56, 58).

Wikle, Christopher K. and Mevin B. Hooten (Nov. 1, 2010). "A general science-based framework for dynamical spatio-temporal models". In: *TEST* 19.3, pp. 417–451. URL: https://doi.org/10.1007/s11749-010-0209-z (cit. on p. 9).

Yin, Yuan, Arthur Pajot, Patrick Gallinari, and Emmanuel de Bézenac (Aug. 2019). "Unsupervised Inpainting for Occluded Sea Surface Temperature Sequences". In: *Climatinformatics Workshop* (cit. on pp. 5, 66).

Yu, Fisher, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao (2015). "LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop". In: *CoRR* abs/1506.03365. URL: http://arxiv.org/abs/1506.03365 (cit. on pp. 67, 86).

Yu, Jason J., Adam W. Harley, and Konstantinos G. Derpanis (2016). "Back to Basics: Unsupervised Learning of Optical Flow via Brightness Constancy and Motion Smoothness". In: *Computer Vision – ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III*. Ed. by Gang Hua and Hervé Jégou. Cham: Springer International Publishing, pp. 3–10. URL: https://doi.org/10.1007/978-3-319-49409-8_1 (cit. on p. 12).

Yu, Jiahui, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang (2018). "Generative image inpainting with contextual attention". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5505–5514 (cit. on p. 24).

Zhang, Han, Ian J. Goodfellow, Dimitris N. Metaxas, and Augustus Odena (2018). "Self-Attention Generative Adversarial Networks". In: *CoRR* abs/1805.08318. URL: http://arxiv.org/abs/1805.08318 (cit. on pp. 21, 85).

Zhu, Jun-Yan, Taesung Park, Phillip Isola, and Alexei A. Efros (Mar. 2017a). "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks". In: *arXiv:1703.10593 [cs]*. URL: http://arxiv.org/abs/1703.10593 (cit. on pp. 21, 22).

Zhu, Jun-Yan, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A. Efros, Oliver Wang, and Eli Shechtman (Nov. 2017b). "Toward Multimodal Image-to-Image Translation". In: *arXiv:1711.11586 [cs, stat]*. URL: http://arxiv.org/abs/1711.11586 (cit. on pp. 22, 79, 81, 85, 99).