Representation Learning and Deep Generative Modeling in Dynamical Systems

Jean-Yves Franceschi

Sorbonne Université — 21 rue de l'École de Médecine, 75006 Paris
École Doctorale Informatique, Télécommunications et Electronique de Paris

Thèse de doctorat de Sorbonne Université

Apprentissage de représentations et modèles génératifs profonds dans les systèmes dynamiques

Representation Learning and Deep Generative Modeling in Dynamical Systems

Jean-Yves Franceschi

Soutenue publiquement pour obtenir le grade de
docteur en Informatique de Sorbonne Université
le 14 février 2022 devant le jury composé de :

M. Xavier Alameda-Pineda	Rapporteur
M. Alexandre Gramfort	Rapporteur
Mme Catherine Achard	Examinatrice
Mme Camille Couprie	Examinatrice
M. Sylvain Lamprier	Directeur de thèse
M. Patrick Gallinari	Directeur de thèse

Dédié à mes parents.

Résumé

Le récent essor de l'apprentissage profond, qui s'est d'abord expliqué par ses divers succès dans l'apprentissage supervisé avec par exemple la classification d'images, trouve également sa source dans les nombreuses avancées scientifiques qu'il a permises en apprentissage non supervisé, plus particulièrement en termes d'apprentissage de représentations et de modèles génératifs. Ces percées contribuent au développement constant des systèmes autonomes acquérant leurs capacités de manière auto-supervisée, car elles leur permettent d'appréhender leur environnement. Dans leur grande majorité, ces progrès ont cependant été obtenus sur des données textuelles et visuelles dont l'évolution à travers le temps est moins étudiée et leur confère une certaine complexité. C'est pourquoi, malgré des efforts de recherche précoces et soutenus dans cette direction, les données temporelles demeurent un défi pour l'apprentissage non supervisé, surtout concernant les données spatio-temporelles complexes telles que les vidéos. Compte tenu de leur importance pour l'automatisation croissante de multiples tâches comme la conduite dans un environnement en perpétuelle évolution, de plus en plus de travaux s'intéressent à cette problématique.

Dans cette thèse, nous nous inscrivons dans cette tendance en étudiant et améliorant plusieurs aspects de la temporalité et des systèmes dynamiques dans les réseaux de neurones profonds pour l'apprentissage non supervisé de représentations et de modèles génératifs. Dans la première partie de notre travail, nous présentons une méthode générale d'apprentissage de représentations non supervisée pour les séries temporelles prenant en compte les besoins pratiques d'efficacité et de flexibilité de telles techniques. Dans un second temps, nous nous intéressons plus spécialement à l'apprentissage pour les séquences structurées de nature spatio-temporelle, couvrant les vidéos et phénomènes physiques. Nous y montrons la corrélation entre la découverte de représentations pertinentes et séparables d'un côté, et de l'autre la fabrique de modèles prédictifs performants sur ces données. Par ce biais, nous soulignons le rôle crucial des équations différentielles pour la modélisation et le plongement adaptés de ces suites temporelles dans les espaces latents de modèles génératifs profonds. Enfin, nous analysons plus généralement dans une troisième partie l'un des modèles génératifs les plus employés, les réseaux antagonistes génératifs, à travers le prisme des systèmes dynamiques en considérant l'évolution temporelle des réseaux impliqués pendant leur entraînement. Leur dynamique pouvant être écrite sous la forme d'une équation différentielle grâce à la théorie des réseaux neuronaux à largeur infinie, nous en déduisons une meilleure compréhension théorique et empirique de ce type de modèle génératif.

Abstract

The recent rise of deep learning, first supported by various supervised learning successes on e.g. image classification, has also been motivated by numerous scientific breakthroughs in unsupervised learning, particularly regarding representation learning and generative modeling. These advances have participated in the development of autonomous, knowledgeable and self-supervised systems, since they contribute to the ability of these systems to understand their environment. However, most of these achievements have been obtained on image or text data, whose evolution through time is less considered and confers to the considered tasks a new layer of complexity. That is why, despite early and steady research in this direction, temporal data, and especially complex spatiotemporal data like videos, remain challenging on unsupervised tasks for current methods. Given their importance for autonomous systems such as self-driving vehicles to adapt in a constantly evolving environment, these challenges have been actively investigated in a growing body of work.

In this thesis, we follow this line of work and simultaneously investigate and improve several underexplored aspects of temporality and dynamical systems in deep unsupervised representation learning and generative modeling. In the first part of this work, we present a general-purpose deep unsupervised representation learning method for time series tackling scalability and adaptivity issues arising in practical applications. We then further study in a second part representation learning for sequences by focusing on structured and stochastic spatiotemporal data: videos and physical phenomena. We show in this context that performant temporal generative prediction models help to uncover meaningful and disentangled representations, and conversely. We highlight to this end the crucial role of differential equations in the successful modeling and embedding of these natural sequences within sequential generative models. Finally, we more broadly analyze in a third part a popular class of generative models, generative adversarial networks, under the scope of dynamical systems. We study the evolution of the involved neural networks with respect to their training time by describing it with a differential equation thanks to the theory of infinite-width neural networks, allowing us to gain novel theoretical and empirical insights on this generative model.

Remerciements

Arrivé au terme de ce doctorat, je réalise que nombreux sont ceux qui m'ont accompagné, soutenu et aidé pendant cette période. Il m'est impossible de tous les nommer ici, mais je tiens à les assurer de ma plus grande reconnaissance.

Je remercie tout d'abord les membres du jury pour avoir accepté d'évaluer mon travail : Catherine Achard, Camille Couprie, et en particulier Xavier Alameda-Pineda et Alexandre Gramfort pour leurs rapports ayant dû être rédigés dans des délais restreints.

Je tiens à exprimer toute ma reconnaissance à mes encadrants et directeurs de thèse Sylvain et Patrick : j'ai tant appris à leur côté, et j'ai particulièrement apprécié discuter avec et travailler auprès d'eux ; merci à eux de m'avoir offert cette magnifique opportunité et guidé pendant cette période tout en me laissant une grande liberté de choix sur mes travaux. J'ai également bénéficié de l'aide précieuse de tous mes formidables coauteurs qui m'ont beaucoup apporté et sans qui cette thèse serait bien plus courte, avec, dans l'ordre chronologique, Aymeric, Martin, Edouard, Mickaël, Jérémie, Emmanuel et Ibrahim. Merci à Nadine et Christophe, dont le soutien sans faille nous a permis de travailler sereinement.

Au-delà de ces collaborations directes, les nombreux membres permanents et non-permanents de l'équipe MLIA me furent tous d'une aide précieuse. Je remercie chaleureusement tous ceux que j'ai pu côtoyer pendant ces 42 mois : vous avez été comme une deuxième famille pendant ce temps précieux, en particulier pendant les périodes de restrictions sanitaires où les autres liens sociaux furent réduits à portion congrue. Merci à ceux avec qui j'ai partagé mon bureau, des repas, des soirées et de grandes parties de babyfoot !

Je suis conscient que ce travail arrive au terme de plusieurs années d'études pendant lesquelles j'ai bénéficié d'un enseignement et accompagnement de grande qualité, que ce soit jusqu'au baccalauréat, en CPGE ou à l'ENS de Lyon. J'ai une pensée particulière pour Marc de Falco, Luc Albert et Omar Fawzi qui m'ont donné ce goût de l'informatique et de la recherche ; merci aussi à mes camarades de promotion, auprès desquels j'ai beaucoup appris.

J'ai également été merveilleusement accompagné par mes amis hors du travail, dont la compagnie fut d'un grand soutien quel que soit le contexte. Sans pouvoir être exhaustif, je pense à Jean-Baptiste, Rémi, Daniel, Anthony, Pierre-Jean, Fabrice, Clément, Lucie, Arthur, Rémy, Maverick, Raphaël, Andreas, Bojana, Prakhar, Aymeric.

Enfin, je dois exprimer ma plus profonde gratitude à ma famille, sans qui je n'aurai rien pu faire. Là encore sans exhaustivité, merci à Laurent, Hélène, Emilie, Marie-Jeanne, Robert, Huguette, Clément, Quentin. Merci à ma grand-mère. Et merci à mes parents, qui ont fait ce que je suis et à qui je dois tout.

Contents

List of Figures

List of Tables

List of Acronyms

CNN	Convolutional Neural Network
DTW	Dynamic Time Warping
ELBO	Evidence Lower Bound
FVD	Fréchet Video Distance
GAN	Generative Adversarial Network
GP	Gaussian Process
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
IPM	Integral Probability Metric
KLD	Kullback-Leibler Divergence
LPIPS	Learned Perceptual Image Patch Similarity
LSGAN	Least Squares GAN
LSTM	Long Short-Term Memory
MLP	Multilayer Perceptron
MMD	Maximum Mean Discrepancy
MSE	Mean Squared Error
NLP	Natural Language Processing
NTK	Neural Tangent Kernel
ODE	Ordinary Differential Equation
PDE	Partial Differential Equation
PSNR	Peak Signal-to-Noise Ratio
RBF	Radial Basis Function
RKHS	Reproducing Kernel Hilbert Space
RNN	Recurrent Neural Network
SDE	Stochastic Differential Equation
SSIM	Structured Similarity
SVM	Support-Vector Machine
VAE	Variational Autoencoder
VRNN	Variational Recurrent Neural Network
WGAN	Wasserstein GAN

Part I.

Motivation

We motivate and contextualize our work in this part by firstly introducing in Chapter 1 the scientific context of this thesis and summarizing our contributions, and by secondly exposing in Chapter 2 the technical background and state of the literature for the notions studied throughout this document.

Chapter 1.

Introduction

1.1. Context

Within the broad field of Artificial Intelligence, machine learning aims at designing systems that automatically improve based on observational data (Russell and Norvig, 2020, Part V). Extending and amplifying the development of machine learning that began two decades ago, the rise of deep learning, i.e. the use of deep artificial neural networks usually optimized via gradient descent techniques (Goodfellow, Y. Bengio, and Courville, 2016), has been a pivotal element behind numerous scientific and technologic breakthroughs since 2012 when the neural network AlexNet (Krizhevsky, Sutskever, and Hinton, 2012) won the ImageNet Large Scale Visual Recognition Challenge (Russakovsky et al., 2015).

Since then, deep learning has benefited from extensive computational improvements with Graphics Processing Units (GPUs) and Tensor Processing Units (Jouppi et al., 2017), software creation (Abadi et al., 2016; Paszke et al., 2019) and the availability of large-scale datasets (Goodfellow, Y. Bengio, and Courville, 2016, Chapter 1). From this, researchers could rapidly achieve significant advances in many domains such as computer vision (Szegedy et al., 2017), Natural Language Processing (NLP) (Yonghui Wu et al., 2016) and games (Silver, A. Huang, et al., 2016) in a supervised learning context. Many of these advances illustrate the singular ability of deep neural networks to learn representations and decision-making systems from raw data.

Unsupervised Representation Learning and Generative Modeling

Nonetheless, the need for machine learning models that could learn with no or little human supervision has emerged, because of the dataset size increase coupled with the limits and cost of human intervention required for creating such datasets (M. Chen, 2020). This need is especially relevant when considering the current development of autonomous, knowledgeable and self-learning systems in technological domains, e.g. autonomous driving and automated human assistance. It even becomes the central issue when it comes to creating models able to outperform human abilities, by definition. Several domains, including few-shot, weakly supervised, transfer and reinforcement learning, address this challenge.

Among them, the broad field of unsupervised learning deals with machine learning models in the absence of human annotations. Research in this direction has allowed the community to push back the limits of the so-called intelligent systems, find new applications and sometimes even improve or help supervised methods (Silver, Hubert,

et al., 2017; Lample et al., 2018; T. Brown et al., 2020; Van Gansbeke et al., 2020). In this regard, the growth of deep learning has been crucial, thanks to now standard and popular techniques: unsupervised representation learning and deep generative models.

Both techniques have indeed shown their relevance by driving many of the scientific and technological advances that deep learning has made possible. On the one hand, unsupervised representation learning has for example been widely leveraged through pretrained (then used for downstream tasks) or dedicated models: e.g., word2vec and BERT in NLP (Mikolov, Sutskever, et al., 2013; Devlin et al., 2019), or the ones of Finn, Goodfellow, and Levine (2016) and Bachman, Hjelm, and Buchwalter (2019) in computer vision and robotics. On the other hand, deep generative models, by learning to imitate real-world distributions and producing new data points, have also provided the field of unsupervised learning with numerous successful applications, such as data augmentation (Antoniou, Storkey, and Edwards, 2017), generating realistic images (Karras, Laine, Aittala, et al., 2020), aligning distributions without supervision (J.-Y. Zhu et al., 2017; Lample et al., 2018) and representation learning (Kingma and Welling, 2019).

Temporal Data

However, most of these recent advances deal with text and image data, leaving temporal data behind. Admittedly, many achievements have also been attained on this data type, whether for unsupervised learning or other settings (Schmidhuber, 2015). They have been mainly based on longstanding sequential architectures called Recurrent Neural Networks (RNNs), like the Long Short-Term Memory units (LSTMs) of Hochreiter and Schmidhuber (1997). These networks have been widely used when it comes to modeling temporal data such as videos, speech, or more generally time series. Nonetheless, apart from such general-purpose networks, unsupervised learning advances for temporal data have largely remained compartmentalized, with innovations in some domains being scarcely transferred to other ones. For instance, while the non-RNN WaveNet has been successfully operated for speech synthesis (van den Oord, Yazhe Li, Babuschkin, et al., 2018), it has not been extensively adopted for other types of data. Moreover, many improvements concerning learning on videos have leveraged earlier innovations in computer vision for images (M. Mathieu, Couprie, and LeCun, 2016; Kalchbrenner et al., 2017; Kumar et al., 2020). Transformers (Vaswani et al., 2017), after their successes in NLP and computer vision, are increasingly used for temporal data as well, e.g. for videos (Weissenborn, Täckström, and Uszkoreit, 2020) and audio (N. Li et al., 2019), but remain distanced by RNNs for this type of data.

Such innovation delay and complexity are not surprising given the complex nature of temporal data. Temporality indeed brings an additional layer of complexity compared to already challenging static data. Another particularity of time series is their diversity: from daily-life information (e.g., trajectories in an environment) to industrial applications (sensor data, videos for autonomous driving) by way of medical data (electroencephalograms, vital signs), many applications of artificial intelligence involve time by nature. In NLP as well, the evolution through time of corpora may be relevant for some applications (Delasalles, Lamprier, and Denoyer, 2019). In a nutshell, handling temporal data is of crucial importance in order to create autonomous and intelligent

systems in a constantly evolving environment (Nagabandi et al., 2019).

This omnipresence, complexity and necessity thus make temporal data an important subject of study in the deep learning community, especially for representation learning and generative modeling in an unsupervised learning context. Given the diversity of sequential data, the development of general-purpose principled models for handling temporality is essential for the community. While RNNs have fulfilled this need as they have been for a long time a fundamental building block of most temporal models, there have been multiple trends in recent years to discover more performant and convenient temporal models.

Dynamical Systems

Among them, a recent research direction consists in modeling time series as dynamical systems, in the physical meaning of the term, i.e. with an evolution described by a differential equation. First considered when injecting prior knowledge in the modeling of physical phenomena (Pajot, 2019), the use of neural dynamical systems has then spread to handle the temporal evolution of more general data since the seminal article of R. T. Q. Chen, Rubanova, et al. (2018). One of the advantages of relying on differential equations is the extensive mathematical and physical literature on the subject. This allows the community to leverage strong preexisting results in order to propose novel and performant models, intrinsically presenting many advantages over RNNs. Besides these new potential developments, and for the same reasons, the theory of dynamical systems can also serve as a powerful tool to analyze, instead of the dynamics of the data, the very training dynamics of deep neural networks (Kovachki and Stuart, 2021), thereby better understanding the reasons behind their successes.

1.2. Subject and Contributions of this Thesis

In this thesis, we follow the above-mentioned lines of work by investigating several aspects of temporality in deep unsupervised representation learning and generative modeling. Our contributions tackle novel applications and issues, mainly articulated around the role of dynamical systems in the improvement and understanding of unsupervised deep learning methods.

We more specifically consider three main research directions:

- general-purpose scalable representation learning for time series;
- the combination of generative modeling and representation learning within a dynamical system framework for the forecasting of complex spatiotemporal data such as natural videos and physical phenomena;
- the theoretical and empirical study of the training dynamics of a standard generative model through the lens of differential equations.

We introduce and summarize our contributions in the following, and then detail the organization of this document.

1.2.1. General-Purpose Unsupervised Representation Learning for Time Series

While supervised learning and forecasting for time series are active and profuse research directions, unsupervised representation learning for this data type remains an under-explored problem. Yet, developing general-purpose unsupervised methods for time series is important with regards to the scarcity and cost of acquiring human-labeled data in most applications, in particular those involving industrial and real-life data. The challenging and noisy nature of real-life time series also makes it preferable for representation learning methods to apply to series of unequal and high lengths. Still, existing unsupervised representation learning methods remain limited with respect to these considerations, besides lacking strong and thorough experimental evaluation.

Tackling these issues, we propose in this thesis a general-purpose unsupervised scalable representation learning method that can handle time series of unequal and high lengths. To this end, we train a deep neural network encoder outputting a fixed-length representation regardless of the size of the input time series thanks to a novel triplet loss relying on time-based positive and negative sampling. The efficiency and flexibility of the chosen encoder, based on dilated convolutions, coupled with the triplet loss requiring no decoder, ensure the generality and scalability of the proposed method. We then assess the quality of learned representations on downstream tasks on standard datasets, thereby showing their transferability and general applicability across different data domains and tasks.

This contribution, detailed in Part II of this document, led to the following publication in an international conference.

> Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi (2019). "Unsupervised Scalable Representation Learning for Multivariate Time Series". In: *Advances in Neural Information Processing Systems*. Ed. by Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily Fox, and Roman Garnett. Vol. 32. Curran Associates, Inc., pp. 4650–4661.

1.2.2. Dynamical Systems and Representation Learning for Complex Spatiotemporal Data

Moving on from general-purpose representation learning, we then more particularly study representation learning for complex structured spatiotemporal data. The latter arise in large-scale applications involving the observation of moving human subjects, objects and physical entities; typical examples of such data include videos and physical phenomena. They still constitute a challenge for neural networks due to their complexity and the need for high computational power to handle them, thus motivating advances that could improve existing models with reasonable resources requirements.

We consider two main types of data: videos and physical phenomena. Videos have numerous applications with respect to autonomous systems, including robotics and self-driving cars. They necessitate predictive models which should generate realistic images and take into account the inherent stochasticity of the observed phenomena.

The applicability of these models highly depends on their representation learning abilities. Indeed, the latter are essential for downstream tasks like planning and action recognition, allowing autonomous systems to benefit from small-scale representations of the environment. The prediction of physical phenomena, possibly less random but more chaotic, is a recent application field of deep learning that still struggles to achieve results equivalent to more classical prediction methods relying on physical models. Representation learning is especially interesting for the latter to understand the prediction mechanisms of data-driven approaches for partially observable data.

Therefore, in this thesis, we explore representation learning for this type of sequence via generative modeling and forecasting. For both considered applications, we design temporal generative prediction models for spatiotemporal data relying on learning meaningful and disentangled representations. We show that the long-term predictive performance and representation learning abilities of these models mutually benefit from each other. An influential modeling choice in this regard is the inspiration from dynamical systems for the design of the proposed temporal evolution models. More precisely, our models are based on discretizations of differential equations parameterized by neural networks, which we show to be particularly adapted to the learning of continuous-time dynamics typically involved in videos and physical phenomena.

These contributions, developed in Part III of this document, were presented in the hereunder two international conference publications.

> Jean-Yves Franceschi, Edouard Delasalles, Mickaël Chen, Sylvain Lamprier, and Patrick Gallinari (July 2020). "Stochastic Latent Residual Video Prediction". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 3233–3246.

> Jérémie Donà, Jean-Yves Franceschi, Sylvain Lamprier, and Patrick Gallinari (2021). "PDE-Driven Spatiotemporal Disentanglement". In: *International Conference on Learning Representations*.

1.2.3. Study of Generative Adversarial Networks via their Training Dynamics

After highlighting the valuable role of dynamical systems for deep generative predictive models, we then leverage differential equations within a novel theoretical framework to analyze and explain the training dynamics of a popular but still misunderstood generative model: Generative Adversarial Networks (GANs).

We point out a fundamental flaw in previous theoretical analyses of GANs that leads to ill-defined gradients for the discriminator. Indeed, within these frameworks that neglect its architectural parameterization as a neural network, the discriminator is insufficiently constrained to ensure the existence of its gradients. This oversight raises important modeling issues as it makes these analyses incompatible with standard GAN practice using gradient-based optimization. We overcome this problem which impedes a principled study of GAN training, solving it within our framework by taking into account the discriminator's architecture and training.

To this end, we leverage the theory of infinite-width neural networks for the discriminator via its Neural Tangent Kernel (NTK) in order to model its inductive biases as a neural network. We thereby characterize the trained discriminator for a wide range of losses by expressing its training dynamics with a differential equation. From this, we establish general differentiability properties of the network that are necessary for a sound theoretical framework of GANs, making ours closer to GAN practice than previous analyses.

Thanks to this adequacy with practice, we gain new theoretical and empirical insights about the generated distribution's flow during training, advancing our understanding of GAN dynamics. For example, we find that, under the integral probability metric loss, the generated distribution minimizes the maximum mean discrepancy given by the discriminator's NTK with respect to the target distribution. We empirically corroborate these results via a publicly released analysis toolkit based on our framework, unveiling intuitions that are consistent with current GAN practice and opening new perspectives for better and more principled GAN models.

This contribution, explained in Part IV of this document, corresponds to the following preprint and is currently under review at an international conference.

> Jean-Yves Franceschi, Emmanuel de Bézenac, Ibrahim Ayed, Mickaël Chen, Sylvain Lamprier, and Patrick Gallinari (2021). "A Neural Tangent Kernel Perspective of GANs". In: arXiv: 2106.05566.

1.2.4. Outline of this Thesis

This document is organized as follows. Chapter 2 explains the state of the literature and the necessary background for the exposition of our contributions. The latter are presented in Chapters 3 to 6: Chapter 3 describes the proposed unsupervised representation learning method for time series, Chapters 4 and 5 respectively introduce our video prediction model and our predictive spatiotemporal disentangling method, and Chapter 6 details our analysis of GAN training dynamics. Finally, Part V, with Chapters 7 and 8, concludes this document with a discussion of the perspectives offered by the exposed contributions. An appendix containing supplementary material for Chapters 3 to 6 is given in Appendices A to D.

Chapter 2.

Background and Related Work

In this chapter, we expose and contextualize the principal notions that are employed and explored in the rest of this document. We first address in Section 2.1 the architectural choices to make neural networks handle sequential data, from standard RNNs to neural differential equations. We then briefly expose in Section 2.2 the state of the art of representation learning for sequences, made possible by these sequential architectures. We finally introduce in Section 2.3 the main deep generative models considered in the literature and how they may be adapted to sequential data.

2.1. Neural Architecture for Sequence Modeling

We review in this section the various neural network architectures designed to deal with sequential inputs and outputs. We begin this review by introducing RNNs in Section 2.1.1, and then present temporal models based on differential equations in Section 2.1.2 as one of the main investigated notions in this manuscript. We finally broach other types of sequential architectures in Section 2.1.3.

2.1.1. Recurrent Neural Networks

RNNs constitute a broad class of models introduced several decades ago (Rumelhart, Hinton, and R. J. Williams, 1986). They have been widely used by the community for many applications (Greff et al., 2017), in fields as diverse as e.g. video prediction (Srivastava, Mansimov, and Salakhudinov, 2015) and language modeling (Mikolov, Karafiát, et al., 2010).

2.1.1.1. Principle

The typical basic RNN can be recursively described as follows (Goodfellow, Y. Bengio, and Courville, 2016, Chapter 10), with sequential inputs $\boldsymbol{x} = (\boldsymbol{x}_t)_{t\in\mathbb{N}}$, outputs $\boldsymbol{y} = (\boldsymbol{y}_t)_t$ and hidden states $\boldsymbol{h} = (\boldsymbol{h}_t)_t$:

$$\boldsymbol{h}_t = \tanh\big(W_{\mathrm{p}}\boldsymbol{x}_t + W_{\mathrm{h}}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{\mathrm{h}}\big), \tag{2.1}$$

$$\boldsymbol{y}_t = W_{\mathrm{o}}\boldsymbol{h}_t + \boldsymbol{b}_{\mathrm{o}}. \tag{2.2}$$

Here, weight matrices W_{p}, W_{h}, W_{o} and bias vectors $\boldsymbol{b}_{\mathrm{h}}$, $\boldsymbol{b}_{\mathrm{o}}$ are the RNN parameters. The first hidden state may be initialized depending on modeling choices, for instance

to the null vector. Note that $\boldsymbol{x}_t$ can also depend on the previous outputs, for example with $\boldsymbol{x}_t = \boldsymbol{y}_{t-1}$ when considering an autoregressive prediction task. This formalization, with shared parameters through time, allows this architecture to handle sequences of arbitrary length.

This RNN may be extended by stacking recurrent transformations of Equation (2.1) over $L \in \mathbb{N}^*$ layers, with layer-dependent hidden states $\boldsymbol{h}^{(l)}$ for layers $l \in [\![1, L]\!]$, and $\boldsymbol{h}^{(0)} = \boldsymbol{x}$:

$$\boldsymbol{h}_t^{(l)} = \tanh\Big(W_{\mathrm{p}}^{(l)} \boldsymbol{h}_t^{(l-1)} + W_{\mathrm{h}}^{(l)} \boldsymbol{h}_{t-1}^{(l)} + \boldsymbol{b}_{\mathrm{h}}^{(l)}\Big). \tag{2.3}$$

Nonetheless, even a shallow RNN of the type of Equation (2.1) is, in theory, a powerful sequential model, as it can compute any function also computable by a Turing machine (Siegelmann and E. D. Sontag, 1994).

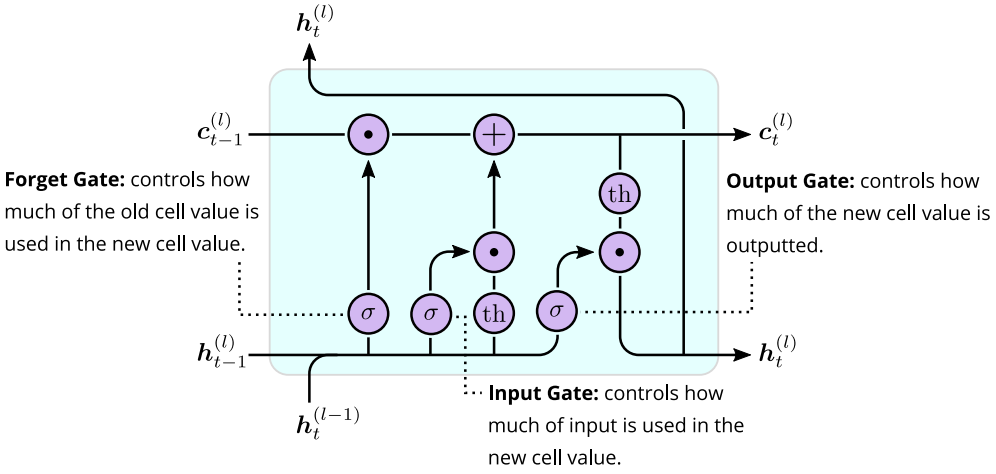
2.1.1.2. Refinements

Despite the theoretical power of their architectures, standard RNNs may be challenging to learn. Because of their recurrent nature, their optimization is hindered by vanishing and exploding gradients, thereby making their learning of long-term dependencies in the data slow or complex (Goodfellow, Y. Bengio, and Courville, 2016, Chapter 10). To circumvent this issue and improve the quality of RNNs on other aspects, several solutions have been proposed.

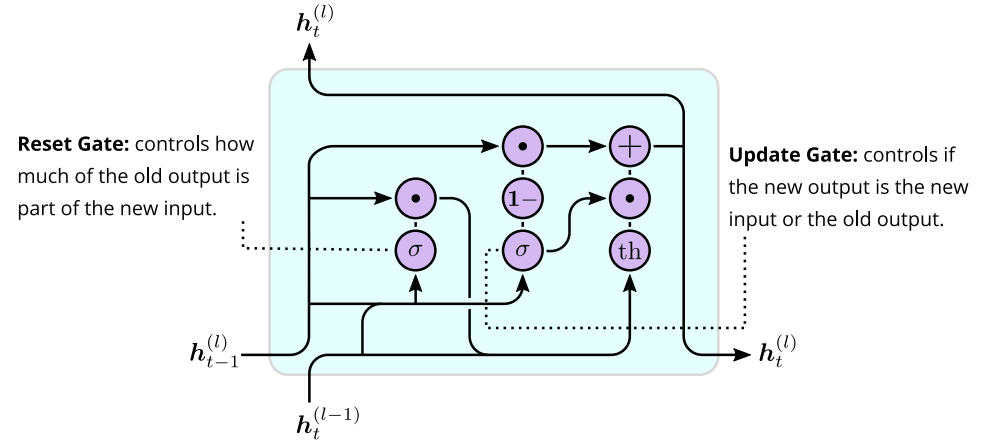
A widely adopted solution is the introduction of gating mechanisms, which allow recurrent networks to accumulate information like in Equation (2.1) but also forget it when it is not needed anymore. This is the principle of LSTMs, as proposed by Hochreiter and Schmidhuber (1997), refined by Gers, Schmidhuber, and Cummins (2000) and popularized in its modern form by Graves and Schmidhuber (2005), which replace the recurrent mechanism of Equation (2.3) with:

$$\begin{aligned}
\boldsymbol{i}_t^{(l)} &= \sigma\Big(W_{\mathrm{i,p}}^{(l)} \boldsymbol{h}_t^{(l-1)} + W_{\mathrm{i,h}}^{(l)} \boldsymbol{h}_{t-1}^{(l)} + \boldsymbol{b}_{\mathrm{i}}^{(l)}\Big), \\
\boldsymbol{f}_t^{(l)} &= \sigma\Big(W_{\mathrm{f,p}}^{(l)} \boldsymbol{h}_t^{(l-1)} + W_{\mathrm{f,h}}^{(l)} \boldsymbol{h}_{t-1}^{(l)} + \boldsymbol{b}_{\mathrm{f}}^{(l)}\Big), \\
\boldsymbol{g}_t^{(l)} &= \tanh\Big(W_{\mathrm{g,p}}^{(l)} \boldsymbol{h}_t^{(l-1)} + W_{\mathrm{g,h}}^{(l)} \boldsymbol{h}_{t-1}^{(l)} + \boldsymbol{b}_{\mathrm{g}}^{(l)}\Big), \\
\boldsymbol{o}_t^{(l)} &= \sigma\Big(W_{\mathrm{o,p}}^{(l)} \boldsymbol{h}_t^{(l-1)} + W_{\mathrm{o,h}}^{(l)} \boldsymbol{h}_{t-1}^{(l)} + \boldsymbol{b}_{\mathrm{o}}^{(l)}\Big), \\
\boldsymbol{c}_t^{(l)} &= \boldsymbol{f}_t^{(l)} \odot \boldsymbol{c}_{t-1}^{(l)} + \boldsymbol{i}_t^{(l)} \odot \boldsymbol{g}_t^{(l)}, \\
\boldsymbol{h}_t^{(l)} &= \boldsymbol{o}_t^{(l)} \odot \tanh \boldsymbol{c}_t^{(l)},
\end{aligned} \tag{2.4}$$

where $\boldsymbol{i}$, $\boldsymbol{f}$ and $\boldsymbol{o}$ are respectively the input, forget and output gates, $\boldsymbol{c}$ is the cell state and σ is the sigmoid function. This model is illustrated in Figure 2.1(a). Intuitively, the cell state $\boldsymbol{c}$ corresponds to a memory state, helping to compute the hidden state $\boldsymbol{h}$ at each time (via the output gate), that is partly propagated through time (the extent of the propagation is controlled by the forget gate), and partly renewed at each time step (through the input gate).



(a) LSTM (Equation (2.4)).



(b) GRU.

Figure 2.1.: Illustration of LSTM and GRU recurrent cells. σ and th respectively denote the sigmoid and tanh functions. Inputs to non-linearities are first transformed by a linear layer. Diagrams are adapted from Madsen (2019), licensed under Creative Commons Attribution CC-BY 4.0.

While LSTMs have become a standard recurrent network that most often outperforms simpler RNNs, other variants and gated networks have been proposed and investigated (Greff et al., 2017). One of the most commonly employed alternatives is the Gated Recurrent Unit (GRU) of Cho et al. (2014), schematically illustrated in Figure 2.1(b), which removes the cell states and merges the forget and input gates of the LSTM. Besides gating mechanisms, other orthogonal axes of improvement have been explored in the literature.

Bidirectional RNNs, for instance, that combine two parallel recurrent networks, one processing inputs forward in time and the other one backward in time, have been proposed by Schuster and Paliwal (1997) to make the output $\boldsymbol{y}_t$ depend on the whole input $\boldsymbol{x}$ and not only the past inputs $\boldsymbol{x}_{\leq t}$. This bidirectional principle can be applied to many RNN variants with success, like LSTMs (Graves and Schmidhuber, 2005). Many other variants of recurrent networks have been suggested (S. Chang et al., 2017; Chung, Ahn, and Y. Bengio, 2017), answering various issues of otherwise standard RNNs.

As presented until now in this document, recurrent networks are mostly fit to vectorial inputs such as multivariate time series or embedded textual data, with internal states being vectors evolving through time. Some adaptation may be needed to make them more efficiently handle other data types (Ganea, Bécigneul, and Hofmann, 2018; Seo et al., 2018). For image-like structured data, Shi et al. (2015) propose the Convolutional LSTM, also named ConvLSTM, by replacing linear operations in Equation (2.4) with convolutional layers and vectorial hidden states and gates with two-dimensional tensors.

Besides such adaptations of existing recurrent networks, another line of research rather considers neural networks as discretizations of continuous-time dynamical models. Recent works have thus developed the links between neural networks and solvers of differential equations to find new sequential architectures that are potentially better adapted to some input data – in our context of Part III, natural data such as videos and physical phenomena.

2.1.2. Neural Differential Equations

We give in this section an overview of differential equations used together with neural networks in the context of temporal modeling, a topic that has been increasingly studied during the extent of the presented thesis.

2.1.2.1. ODEs and PDEs

Ordinary Differential Equations (ODEs) are differential equations whose unknown is a function y of a single real univariate variable – that we consider in this document to be the time variable t – to a Banach space and whose expression involves t, $y(t)$ and its derivatives $\frac{\mathrm{d}^k y}{\mathrm{d}t^k}(t)$ (up to a finite maximal order $n \in \mathbb{N}$, with $k \leq n$), also denoted by $\partial_t^k y_t$. We are mainly interested in explicit ODEs, expressed as:

$$\frac{\mathrm{d}^n y}{\mathrm{d}t^n}(t) = f\left(t, y(t), \frac{\mathrm{d}y}{\mathrm{d}t}(t), \ldots, \frac{\mathrm{d}^{n-1} y}{\mathrm{d}t^{n-1}}(t)\right), \tag{2.5}$$

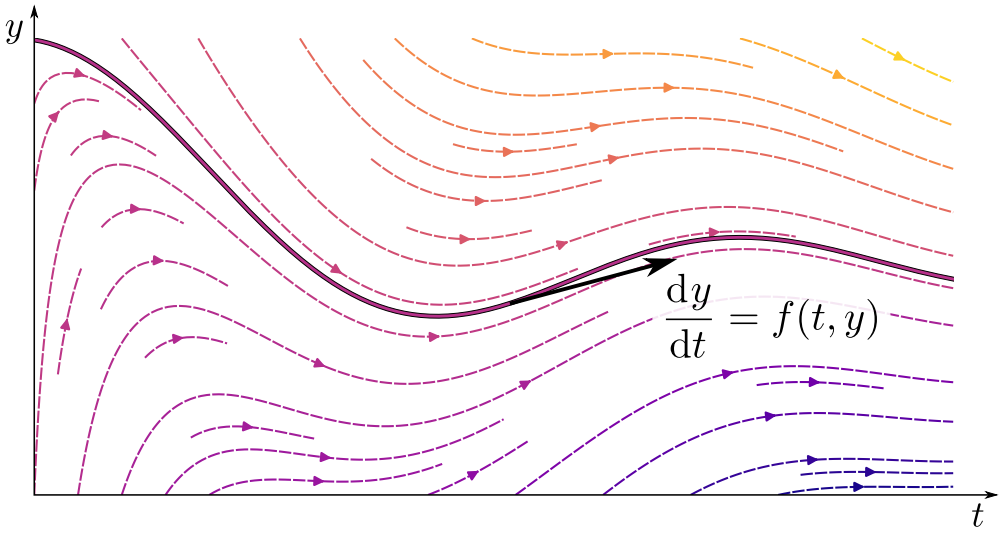


Figure 2.2.: Illustration (in dashed lines) of the continuous flow of an ODE, with a particular solution that is plotted with a solid thicker line. The black arrow represents the tangent to the highlighted solution, which is fully determined by its derivative and initial condition according to variants of the Cauchy-Lipschitz theorem (Demailly, 2006, Chapter V, Section 3.4). In this example, f is defined as $f\colon (t, y) \mapsto \frac{1}{t}(\cos t - y)$, the ODE admitting as solutions over $I = (0, +\infty)$ functions $y_C\colon t \mapsto \frac{C}{t} + \operatorname{sinc} t$ for all $C \in \mathbb{R}$.

where f is a function of n variables, usually defined over an open set (in particular, with t belonging to an open interval I). For the sake of clarity, the latter equation is often abbreviated as:

$$\frac{\mathrm{d}^n y}{\mathrm{d}t^n} = f\left(t, y, \frac{\mathrm{d}y}{\mathrm{d}t}, \ldots, \frac{\mathrm{d}^{n-1} y}{\mathrm{d}t^{n-1}}\right). \tag{2.6}$$

Note that an n-th order explicit ODE can be written as a first-order ODE by grouping y and its first $n - 1$ derivatives into a single n-dimensional $\boldsymbol{y}$. Therefore, we mainly consider in this document without loss of generality first-order ODEs with an arbitrary number of dimensions of the form:

$$\frac{\mathrm{d}\boldsymbol{y}}{\mathrm{d}t} = f(t, \boldsymbol{y}). \tag{2.7}$$

ODEs have various applications in multiple domains such as physics and biology, resulting in a diverse field of study with extensive accumulated knowledge (Polyanin and Zaitsev, 2002). In the context of our work, we are mainly interested in general results about Equation (2.7). More particularly, we often rely on the existence and uniqueness of solutions to ODEs. While there exist more general but weaker results, we leverage the Cauchy-Lipschitz theorem stating that there exists a unique solution $\boldsymbol{y}$

to Equation (2.7) defined over I under any initial condition $\boldsymbol{y}(0) = \boldsymbol{y}_0$, provided that f is continuous with respect to its first input t and Lipschitz-continuous with respect to its second input $\boldsymbol{y}$ (Benzoni-Gavage, 2010, Theorem 5.9). This result implies that Equation (2.7) defines a time-dependent flow, as illustrated in Figure 2.2.

While we lean on ODEs in most of this document, we are also interested in Partial Differential Equations (PDEs), which appear in many scientific applications as well. They are inherently more complex than ODEs as their unknown u is a function of multiple variables, typically time t and spatial coordinates x, y, etc. Consequently, existence and uniqueness theorems for PDEs are not as universal as for ODEs, but there exist multiple resolution techniques for specific types of PDEs (Le Dret and Lucquin, 2016).

2.1.2.2. Differential Equations and Neural Networks

The numerous applications of differential equations motivate their potential combination with neural networks. As recently popularized by R. T. Q. Chen, Rubanova, et al. (2018) and inspired by control theory, ODEs may be leveraged to model the temporal evolution of a system given an initial condition by learning them through the parameterization of f in Equation (2.7) as a neural network f_θ:

$$\frac{\mathrm{d}\boldsymbol{y}}{\mathrm{d}t} = f_\theta(t, \boldsymbol{y}). \tag{2.8}$$

This is made possible thanks to the existence and uniqueness of properties given by the Cauchy-Lipschitz theorem since a neural network f_θ with standard architecture is Lipschitz-continuous. This results in a continuous-time model which may be used to model latent dynamics and that is more adapted to physical phenomena than standard RNNs, given its grounding in differential equations.

A more rarely used framework, although already present in the early continuous-time RNNs (Pineda, 1988; Almeida, 1990; Funahashi and Nakamura, 1993; Hasani et al., 2021), is to integrate an input x directly into the neural differential equation and not only through its initial condition:

$$\frac{\mathrm{d}\boldsymbol{y}}{\mathrm{d}t} = f_\theta\big(t, \boldsymbol{y}, \boldsymbol{x}(t)\big), \tag{2.9}$$

in which case this system acts as a sequence-to-sequence model like standard RNNs.

Multiple methods to learn such systems have been found. In particular, if one can compute gradients for θ or the initial condition $\boldsymbol{y}_0$ with respect to some learning objective, they can then be used to compute gradients for other parts of the learning system by using standard backpropagation (Rumelhart, Hinton, and R. J. Williams, 1986). The central proposition of R. T. Q. Chen, Rubanova, et al. (2018) to compute these gradients is to solve Equation (2.8) forward in time with any solver which can return an arbitrarily close approximation to the real solution of the ODE; then the associated gradients are computed by finding that they result from another differential equation, which is solved backward in time. This technique called the adjoint state method – from the foundational work of Pontryagin et al. (1962) and Lions (1971), see

also the review of Plessix (2006) – takes advantage of ODEs reversibility and can be seen as a continuous-time version of standard backpropagation (le Cun, 1988). This adjoint method is convenient because it is memory-efficient, similarly to e.g. Gomez et al. (2017), and allows to learn models that are as close as continuous-time as possible. However, it is prone to numerical errors of ODE solvers (Gholaminejad, Keutzer, and Biros, 2019; Zhuang et al., 2020).

Another widespread possibility is to discretize the ODE through a numerical solver and backpropagate through the finite number of operations in the latter, like backpropagation through time for RNNs. When using the simple Euler scheme with step size Δt, this gives:

$$\boldsymbol{y}_{t+\Delta t} = \boldsymbol{y}_t + \Delta t \cdot f_\theta(t, \boldsymbol{y}_t). \tag{2.10}$$

With $\Delta t = 1$ and integer time steps k, one then retrieves an instantiation of residual networks (K. He et al., 2016):

$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + f_{k,\theta}(\boldsymbol{y}_k). \tag{2.11}$$

This correspondence has been noticed by Y. Lu et al. (2018), Haber and Ruthotto (2018), and Ruthotto and Haber (2020) and led authors to propose new architectures by leveraging more involved ODE solving schemes or changing the ODE of Equation (2.8) (Sander et al., 2021). Similarly, Tallec and Ollivier (2018) and De Brouwer et al. (2019) find that GRUs and LSTMs can be seen, in part, as discretizations of ODEs. In that sense, any discretization of a neural-parameterized ODE such as in Equation (2.10) can be interpreted as a recurrent cell defining a new type of recurrent network. Accordingly, B. Chang et al. (2019), Voelker, Kajić, and Eliasmith (2019), and Rusch and Mishra (2021), for example, introduced new recurrent cells based on specific forms of differential equations. Note that, while we study them to model temporal data, neural ODEs have also been applied to static data like in image classification and generation (R. T. Q. Chen, Rubanova, et al., 2018; Dupont, Doucet, and Teh, 2019; Grathwohl et al., 2019). In this case, time t is not bound to the data and is rather a continuous abstraction of the depth of a neural network: inputs are given at time $t = 0$ and outputs are returned at some time $t = T$.

The case of PDEs is more scarcely studied in the literature as it is more difficult to introduce general methods. A number of works have noticed the links between spatial derivatives in PDEs and convolutional networks (Z. Long, Y. Lu, X. Ma, et al., 2018; Z. Long, Y. Lu, and Dong, 2019; Ruthotto and Haber, 2020) due to the relationships between convolutions and finite difference approximation methods, thereby explaining the utility of convolutions even in latent spaces when it comes to predicting complex spatiotemporal phenomena (Shi et al., 2015; Ayed et al., 2020). For specific phenomena, physics-informed deep learning methods may be designed for more accurate predictions grounded in physical knowledge (de Bézenac, Pajot, and Gallinari, 2018; You Xie et al., 2018; Raissi, Perdikaris, and Karniadakis, 2019; R. Wang et al., 2020). In computer vision, the differential equation describing constant illumination in a scene (Horn and Schunck, 1981) has been the founding principles of many methods relying on motion and optical flow methods (D. Sun et al., 2008; Dosovitskiy, Fischer, et al., 2015; Finn, Goodfellow, and Levine, 2016; J. J. Yu, Harley, and Derpanis, 2016).

Finally, even though this is out of the scope of this thesis, we notice an emerging trend of combining Stochastic Differential Equations (SDEs) and neural networks. SDEs complement usual ODEs with Brownian white noise via Wiener processes, and sometimes Poisson processes to incorporate jumps in the dynamics as well. They are the basis of various physics, biology and finance models (Ryder et al., 2018), and an appealing modeling choice thanks to their integration of noise directly in the temporal model. Although they may be challenging to articulate with neural networks, research in this direction has been conducted in a line of works – e.g. non-exhaustively of Ha et al. (2018), Ryder et al. (2018), and X. Li et al. (2020) – since they are particularly adapted to some data types as a stochastic continuous dynamical model.

2.1.2.3. ODEs and Neural Network Optimization

We would like to note in this introduction that the use of ODEs in deep learning is not restricted to neural-parameterized sequence modeling, as it has also been independently leveraged to analyze, and even sometimes improve, the very training dynamics of neural networks with respect to training time.

Let us indeed consider a network f_θ with parameters θ, optimized by plain gradient descent with learning rate λ to minimize the loss function $\mathcal{L}_\theta$. A gradient descent iteration for optimization step k consists in:

$$\theta_{k+1} = \theta_k - \lambda \frac{\partial \mathcal{L}_{\theta_k}}{\partial \theta_k}\,, \tag{2.12}$$

which is the Euler discretization with step size $\Delta t = 1$ of the following ODE:

$$\frac{\mathrm{d}\theta_t}{\mathrm{d}t} = -\lambda \frac{\partial \mathcal{L}_{\theta_t}}{\partial \theta_t}\,, \tag{2.13}$$

where t here denotes the training time.

This usual observation has been used to analyze neural network optimization algorithms (Barakat and Bianchi, 2021), but also standard gradient-based optimization procedures (Belotto da Silva and Gazeau, 2020; Su, S. Boyd, and Candès, 2016; A. A. Brown and Bartholomew-Biggs, 1989); even mini-batch training may be studied through the lens of SDEs. Other ODEs describing the evolution of the parameterized function f_{θ_t} as well as the loss $\mathcal{L}_{\theta_t}$ can then be derived from the description of the parameter evolution through time by Equation (2.13). Jacot, Gabriel, and Hongler (2018) thereby derive from these ODEs the theory of NTKs for infinite-width neural networks, which simplifies these differential equations and their subsequent analysis. We base one of our contributions, in Chapter 6, on this theory and the resulting training ODEs in order to theoretically and empirically study GAN training. Note that numerous authors have also considered ODEs, but never in this specific setting nor with the same generality, to analyze and improve GANs (Mescheder, Nowozin, and A. Geiger, 2017; Nagarajan and Kolter, 2017; Balduzzi et al., 2018; C. Wang, H. Hu, and Y. M. Lu, 2019), with the most recent example (Qin et al., 2020) transforming the Euler discretization of Equation (2.12) into a more involved ODE higher-order solving scheme.

2.1.3. Alternatives

We close this summary of neural architectures for sequential data by mentioning two common alternatives to the approaches described hereinabove.

2.1.3.1. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) (LeCun, Boser, et al., 1989) have been one of the core architectural advances that have driven the many successes of deep learning in computer vision (Krizhevsky, Sutskever, and Hinton, 2012; M. Tan and Le, 2019). Their advantages are numerous (Goodfellow, Y. Bengio, and Courville, 2016, Chapter 9): they implement translation-equivariant functions, learn sparse and local correlations, can be adaptable to varying input dimensions, and are efficiently implementable. Their use, however, is not restricted to static inputs as a temporal dimension can also be included in convolutional operations, and the aforementioned advantages are proven to be useful for temporal applications. While less widespread than RNNs for sequential inputs, CNNs have a long history of applications to sequential data, for example in NLP (Collobert and Weston, 2008; X. Zhang, Zhao, and LeCun, 2015; Gehring et al., 2017) and, in early work labeled as time-delay neural networks, speech recognition (Waibel et al., 1989; Bottou et al., 1990) and videos (Wöhler and Anlauf, 2001). More recently, three-dimensional convolutions, including one temporal and two spatial dimensions, have also been applied to synthesize videos, both in the generator and discriminator networks of GANs (A. X. Lee, R. Zhang, et al., 2018; Y.-L. Chang et al., 2019; Yin et al., 2020).

Close to the issues investigated in Chapter 3, CNNs have also been adopted for time series classification and generation (Cui, W. Chen, and Y. Chen, 2016; Zhiguang Wang, Z. Yan, and Oates, 2017) – cf. the exhaustive review of Ismail Fawaz et al. (2019). Time series generation and forecasting have partly been influenced by the causal CNN architecture, popularized by van den Oord, Dieleman, et al. (2016), and used for instance by Y. Bai, T. Ma, and Risteski (2019), van den Oord, Yazhe Li, Babuschkin, et al. (2018), and Aksan and Hilliges (2019). This architecture implements a sequence-to-sequence function with the peculiar property that the i-th output only depends on the first to i-th inputs, hence the causal attribute. This property is achieved by stacking exponentially dilated convolutions (see e.g. F. Yu and Koltun (2016)). The aforementioned works show that such dilated convolutions help to build networks for sequential tasks outperforming recurrent neural networks in terms of both efficiency and prediction performance. We refer to Chapter 3 for more details on this kind of architecture.

2.1.3.2. Transformers

Transformer networks, proposed by Vaswani et al. (2017) and based on a self-attention mechanism, have proven to be a seminal architecture for textual data. Their introduction, their application to different tasks (Ott et al., 2018; Z. Yang et al., 2019) and their central role in large-scale general-purpose models (Devlin et al., 2019; T. Brown et al., 2020) motivate their swift adoption by the NLP community.

Based on their impressive performance in NLP, the transformer architecture constitutes a good candidate for other applications on temporal data, besides its successful

use on static data in computer vision (Dosovitskiy, Beyer, et al., 2021). Although self-attention is not a sequential operation – it is invariant with respect to the order of its inputs, i.e. permutation-invariant –, transformers include two operations tackling the temporality of the data. Firstly, they combine their input with positional embeddings ensuring that the input of the network takes into account the elements' positions in the sequence. Secondly, the transformer decoder is used in an autoregressive manner, by producing each output at a time conditioned on the previously generated outputs.

To this day, the use of transformers on temporal data has only just begun but already shows promising results. They have been applied and adapted, for instance, to video processing (Weissenborn, Täckström, and Uszkoreit, 2020; Bertasius, H. Wang, and Torresani, 2021; Y.-F. Wu, Yoon, and Ahn, 2021), audio generation (N. Li et al., 2019), time series (Nambiar et al., 2020; N. Wu et al., 2020), dynamical systems (Geneva and Zabaras, 2020), and even reinforcement learning (Lili Chen et al., 2021; Janner, Q. Li, and Levine, 2021). Finally, the computational bottleneck of self-attention, with a quadratic computational cost with respect to the input's length, has been investigated to improve their scalability for long sequences (Katharopoulos et al., 2020).

2.2. Unsupervised Representation Learning for Temporal Data

We succinctly summarize in this section the state of the literature for unsupervised representation learning on temporal data. Given the wide range of this topic, both on the representation learning and temporal data sides, this presentation is not meant to be exhaustive, but rather contextualizes our contributions in the next chapters by highlighting the two main research orientations on this matter: contrastive learning in Section 2.2.1 and autoencoding in Section 2.2.2.

2.2.1. Contrastive Learning

Contrastive learning has been, for some years, the basis for multiple advances in self-supervised representation learning on image and sequential data (T. Wang and Isola, 2020). In opposition to autoencoding methods which require a decoder to train the encoder, contrastive learning removes the need for decoders as it consists in learning representations of data points by comparing them with the representations of others. More specifically, it posits the knowledge of pairs of positive and negative examples. Contrastive learning then aims, as much as possible, at matching the representations of elements of positive pairs – assumed to be similar samples of the training distribution – while dissociating the representations of negative pairs, supposed to be dissimilar elements. This high-level description has been implemented in various forms, usually involving triplet losses (Mikolov, Sutskever, et al., 2013; Balntas et al., 2016; Logeswaran and H. Lee, 2018) originating in the early Siamese networks principle (Bromley et al., 1994).

Usually, this means that a data point $\boldsymbol{x}^{\text{ref}}$ (anchor, or reference) is provided with positive and negative samples, respectively $\boldsymbol{x}^{\text{pos}}$ and $\boldsymbol{x}^{\text{neg}}$. An encoder f_θ with parameters

θ would then be trained using contrastive learning via, for instance, the minimization of a triplet margin loss (Balntas et al., 2016):

$$\max\left(\left\|f_\theta\left(\boldsymbol{x}^{\text{ref}}\right) - f_\theta(\boldsymbol{x}^{\text{pos}})\right\|_2^2 - \left\|f_\theta\left(\boldsymbol{x}^{\text{ref}}\right) - f_\theta(\boldsymbol{x}^{\text{neg}})\right\|_2^2 + \mu, 0\right), \tag{2.14}$$

where $\mu > 0$ is a scalar margin parameter scaling the ideal distance separating positive and negative examples pairs. This specific triplet loss encourages f_θ to satisfy:

$$\left\|f_\theta\left(\boldsymbol{x}^{\text{ref}}\right) - f_\theta(\boldsymbol{x}^{\text{neg}})\right\|_2^2 \geq \left\|f_\theta\left(\boldsymbol{x}^{\text{ref}}\right) - f_\theta(\boldsymbol{x}^{\text{pos}})\right\|_2^2 + \mu, \tag{2.15}$$

in which case its minimum is attained, thereby respecting the preexisting knowledge of positive and negative example pairs.

The challenge of contrastive learning is to have access to some notion of pairwise similarity between samples in order to construct positive and negative pairs, especially in the unsupervised, or self-supervised, setting. For this reason, sequential data like text, video and audio are interesting data types for contrastive learning, because the information provided by the temporality of the sequences can be used to propose performant heuristics of positive and negative pairs selection (van den Oord, Yazhe Li, and Vinyals, 2018). The notion of locality in time often serves to design such heuristics (Hyvarinen and Morioka, 2016; Sermanet et al., 2018; Jansen et al., 2018), similarly to locality in images which can be used in the static case (T. Chen et al., 2020) via random cropping. Intuitively, temporally close elements should have similar representations, and the opposite should hold for elements that are temporally far from each other.

However, this often requires some part of supervision or labeling (Bredin, 2017; Han, W. Xie, and Zisserman, 2020), more particularly to select pairs of positive examples. For instance, Sermanet et al. (2018) learn video representations by relying on the availability of multiple viewpoints for the same clip: frames displaying the same action from different viewpoints are taken as positive examples pairs. Prior to this thesis, the vast majority of contrastive representation learning methods for time series were fully or semi-supervised. In Chapter 3, we tackle this issue and propose a fully unsupervised contrastive representation learning technique focusing on time series.

2.2.2. Learning from Autoencoding and Prediction

Like for static data (Y. Bengio, Courville, and P. Vincent, 2013), representations for sequences can be learned by embedding the data points into a lower-dimensional space through autoencoding techniques. They require the training of a decoder jointly with the encoder, making them potentially more computationally intensive than the previously presented contrastive techniques. However, this frees them from the incentive to take advantage of human supervision, in opposition to contrastive learning.

In the following, we first summarize in Section 2.2.2.1 how autoencoding techniques are adapted to sequential data. We then cover in Section 2.2.2.2 the efforts at learning disentangled representations of temporal inputs within this autoencoding framework.

2.2.2.1. Learning Methods

Autoencoding techniques may solely involve sequence-to-sequence models based on RNNs (Srivastava, Mansimov, and Salakhudinov, 2015; Malhotra et al., 2017; Lyu et al., 2018; Hsieh et al., 2018). They rely on an RNN-based encoder, whose final hidden state serves as a representation of the sequence. This representation is then fed to an RNN-based decoder which reconstructs the encoded sequence. Both networks are usually trained jointly via a standard reconstruction loss, such as the Mean Squared Error (MSE).

Other autoencoding methods rather rely on sequential latent variational models, which are sequential generative models described in Section 2.3.2.3. They differ from standard RNNs by replacing deterministic hidden states with stochastic states and are learned via variational inference. Nevertheless, they also involve autoencoding mechanisms enabling them to learn representations, improved by the regularization properties provided by the characteristics of their generative training. We refer to Section 2.3.2 for a more detailed exposition. Among these models, state-space models, where the representation corresponding to any time step should completely represent the whole system at this moment, are especially advantageous. This has been explored, for instance, by Fraccaro, Kamronn, et al. (2017), Karl et al. (2017), Yingzhen and Mandt (2018), Gregor et al. (2019), and A. X. Lee, Nagabandi, et al. (2020) for various data types such as videos, audio and physical simulations. In this thesis, we leverage such a technique in Chapter 4 with a novel and improved temporal model.

Along the same lines, some other works learn representations by teaching a model to predict the evolution of sequences conditioned by a few past time steps, rather than observing the whole sequences to autoencode them. This approach is chosen in the absence of sequential latent variational models and may benefit the learned representations by constraining them to contain the necessary information in order to predict the unseen future of the sequence (Srivastava, Mansimov, and Salakhudinov, 2015). This choice seems relevant for high-dimensional data, for which the dimensionality reduction performed by representation learning is necessarily more drastic. Examples include Villegas, J. Yang, Hong, et al. (2017) for video prediction, but also several methods tackling physical phenomena which rely on learning to predict for the retrieval of physically meaningful system states (Mrowca et al., 2018; R. T. Q. Chen, Rubanova, et al., 2018; Samuel Greydanus, Dzamba, and Yosinski, 2019; Ayed et al., 2020; Norcliffe et al., 2021). We follow this line of research in Chapter 5 in the context of high-dimensional spatiotemporal data.

2.2.2.2. Disentangled Representations

Seeking disentanglement properties in neural network models has received significant attention in the deep learning community (Y. Bengio, Courville, and P. Vincent, 2013), despite discussions concerning its rigorous definition (Higgins, D. Amos, et al., 2018; Locatello et al., 2019). Disentanglement for temporal data has mostly been approached together with autoencoding techniques. Indeed, it usually aims at separating factors of variation within the data dynamics, which is necessarily modeled via sequential

architectures in autoencoding techniques, in opposition to contrastive methods.

In the sequential case, disentangling the dynamics implies separating it into smaller components which are individually easier to model than if they were combined. This is also the basis of a research direction to improve forecasting models, based on the sequential architectures of Section 2.1 and sometimes on the generative modeling techniques of Section 2.3.2: this is because it simplifies their handling of data dynamics by separating it into distinct building blocks. Several types of disentanglement have been considered in the literature, like distinguishing individual components and their eventual interaction (Hsieh et al., 2018; van Steenkiste et al., 2018), expressing the dynamics at different time scales (Hsu, Yu Zhang, and Glass, 2017; Yingzhen and Mandt, 2018), extracting physical dynamics (Le Guen and Thome, 2020) and constructing structured frame representations (Villegas, J. Yang, Zou, et al., 2017; Minderer et al., 2019; Z. Xu et al., 2019).

A commonly used kind of disentanglement, that is explored in Part III of this thesis, is to extract from the temporal model static features which do not need to be incorporated in the complex sequential network. By entirely removing potentially complex information from temporal models, such disentanglement can drastically decrease the latent dimensionality of sequential generative models. Accordingly, this should reduce the difficulty of learning these temporal models. A particularly interesting application case is videos, for which static components may include visual appearance and background (Vondrick, Pirsiavash, and Torralba, 2016), at least for short sequences. As such, some works have taken advantage of variational inference – like most state-of-the-art static disentanglement methods Locatello et al. (2019) – and adversarial losses, presented in the forthcoming section, to achieve such spatiotemporal disentanglement (Denton and Birodkar, 2017; Villegas, J. Yang, Hong, et al., 2017; Hsieh et al., 2018; Tulyakov et al., 2018), sometimes with data-specific assumptions (Kosiorek et al., 2018; Jaques, Burke, and Hospedales, 2020).

2.3. Deep Generative Modeling

Generative modeling broadly consists in finding a generated distribution α, which can be sampled through a generative model, to approximate a target distribution β that is assumed to be unknown but partially accessible through samples (Jebara, 2008). The generated α is thus a means to artificially generate new samples from β.

A typical application case, which we shall follow in the rest of this chapter, is where β is the underlying distribution of the data p_{data} which is only accessed through the training dataset. Deep generative models typically frame α as a learnable distribution p_θ parameterized by neural network parameters θ. For ease of exposition, we suppose in this section that both p_{data} and p_θ are continuous distributions over some space $\mathcal{X}$ and admit density functions.

We first describe in Section 2.3.1 how different standard classes of deep generative models learn p_θ to approximate p_{data}. We then discuss in Section 2.3.2 how these models have been adapted for sequential data in the literature, in part thanks to the sequential architectures of Section 2.1.

2.3.1. Families of Deep Generative Models

Generative models are traditionally learned by maximizing the log-likelihood of the training data via the model:

$$\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \log p_\theta(\boldsymbol{x}). \tag{2.16}$$

Indeed, doing so is equivalent to minimizing the Kullback-Leibler Divergence (KLD) D_{KL} (Kullback and Leibler, 1951) between p_{data} and p_θ (Goodfellow, 2016, Section 2.1), since:

$$D_{\text{KL}}(p_{\text{data}} \parallel p_\theta) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \log \frac{p_{\text{data}}(\boldsymbol{x})}{p_\theta(\boldsymbol{x})} = -\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \log p_\theta(\boldsymbol{x}) - H(p_{\text{data}}), \tag{2.17}$$

where the entropy $H(p_{\text{data}})$ is constant with respect to θ.

However, this optimization objective requires to compute the generative model's likelihood, which is generally intractable when parameterizing it with neural networks. This is because deep generative models, in order to gain modeling capacity, only implicitly define this likelihood. Indeed, they usually specify transformations from a sampled $\boldsymbol{z}$ of a latent space $\mathcal{Z} = \mathbb{R}^d$ to $\mathcal{X}$ with complex neuronal transformations, making the likelihood intractable in the general case. More formally, the model generates samples $\boldsymbol{x}$ by first sampling $\boldsymbol{z} \in \mathcal{Z}$ from a prior $p(\boldsymbol{z}) = p_{\boldsymbol{z}}$, and then sampling $\boldsymbol{x}$ conditionally to $\boldsymbol{z}$ via $p_\theta(\boldsymbol{x} \mid \boldsymbol{z})$, yielding:

$$p_\theta(\boldsymbol{x}) = \int_z p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) p(\boldsymbol{z}) \, \mathrm{d}\boldsymbol{z}. \tag{2.18}$$

In this general framework, the prior $p_{\boldsymbol{z}}$ is simple – typically, a standard Gaussian – and $p_\theta(\boldsymbol{x} \mid \boldsymbol{z})$ is a distribution $\mathcal{G}$ (e.g. Gaussian or Dirac) whose parameters (e.g. location and variance) are computed via a generator neural network g_θ:

$$p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) = \mathcal{G}\big(g_\theta(\boldsymbol{z})\big) = \mathcal{N}\Big(\mu_{\theta_1}(\boldsymbol{z}), \sigma_{\theta_2}(\boldsymbol{z})^2\Big) \triangleq \mathcal{N}\Big(\mu_{\theta_1}(\boldsymbol{z}), \text{diag}\Big(\sigma_{\theta_2}(\boldsymbol{z})^2\Big)\Big), \tag{2.19}$$

in the factorized Gaussian case. In the Dirac case, this would instead correspond to:

$$\boldsymbol{x} = g_\theta(\boldsymbol{z}). \tag{2.20}$$

As a result, $p_\theta(\boldsymbol{x} \mid \boldsymbol{z})$ can be analytically calculated, but the complexity of the involved transformation makes the marginalization $p_\theta(\boldsymbol{x})$ of $p_\theta(\boldsymbol{x} \mid \boldsymbol{z})$ by $p_{\boldsymbol{z}}$ overly complex to compute.

The complexity of this neural network generator and the resulting intractability of the log-likelihood maximization objective motivate the introduction of the two main generative models described and implemented in this thesis: Variational Autoencoders (VAEs) and GANs. After introducing them in the following, we then briefly address other deep generative models tackling Equations (2.16) and (2.18) by adapting the generator architecture.

2.3.1.1. Variational Autoencoders

The principle of VAEs (Kingma and Welling, 2014; Rezende, Mohamed, and Wierstra, 2014) consists in, instead of directly optimizing the intractable Equation (2.16), maximizing a lower bound of the latter. To this end, another ϕ-parameterized distribution $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$, called the inference or recognition network, is introduced as an approximation of the inaccessible true posterior $p_\theta(\boldsymbol{z} \mid \boldsymbol{x})$. This enables the following derivation:

$$\begin{aligned}\log p_\theta(\boldsymbol{x}) = \mathbb{E}_{z \sim q_\phi(\boldsymbol{z} \mid \boldsymbol{x})} \log p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) - D_{\mathrm{KL}}\big(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \,\big\|\, p(\boldsymbol{z})\big) \\ + D_{\mathrm{KL}}\big(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \,\big\|\, p_\theta(\boldsymbol{z} \mid \boldsymbol{x})\big).\end{aligned} \tag{2.21}$$

Since KLD is non-negative, the following so-called Evidence Lower Bound (ELBO) appears:

$$\log p_\theta(\boldsymbol{x}) \geq \underbrace{\mathbb{E}_{\boldsymbol{z} \sim q_\phi(\boldsymbol{z} \mid \boldsymbol{x})} \log p_\theta(\boldsymbol{x} \mid \boldsymbol{z})}_{\text{reconstruction term}} - \underbrace{D_{\mathrm{KL}}\big(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \,\big\|\, p(\boldsymbol{z})\big)}_{\text{KLD term}} \triangleq \mathcal{L}_{\mathrm{ELBO}}(\theta, \phi). \tag{2.22}$$

The latter lower bound is then maximized with respect to both θ and ϕ.

This is possible thanks to a proper choice of distribution $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$. Canonically, the latter is chosen as a factorized Gaussian distribution whose mean and variance are computed by encoder networks μ_{ϕ_1}, σ_{ϕ_2} from the data point $\boldsymbol{x}$:

$$q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) = \mathcal{N}\Big(\mu_{\phi_1}(\boldsymbol{x}), \sigma_{\phi_2}(\boldsymbol{x})^2\Big). \tag{2.23}$$

In this case, and when the prior is Gaussian as well, the KLD term is analytically computable thanks to the closed-form expression of the KLD between Gaussian distributions. Moreover, the reconstruction term can be approximated via Monte-Carlo sampling, with unbiased and low-variance gradient estimates via backpropagation by sampling from $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$ using the reparameterization trick. The latter consists in applying a gradient-tracking transformation to a sample from a constant distribution instead of directly sampling from the learned Gaussian of Equation (2.23) where backpropagation is impossible. $\boldsymbol{z}$ is then sampled as follows:

$$\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, I_d), \qquad \boldsymbol{z} = \mu_{\phi_1}(\boldsymbol{x}) + \boldsymbol{\varepsilon} \odot \sigma_{\phi_2}(\boldsymbol{x}), \tag{2.24}$$

where $\odot$ denotes the Hadamard product.

Interpreting $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$ as an encoding mechanism, coupled with the generator function of Equation (2.19), supports the denomination of VAEs as autoencoders. This is because the reconstruction term of the ELBO of Equation (2.22) favors the successful decoding of $\boldsymbol{z}$ as a representation of $\boldsymbol{x}$ into $\boldsymbol{x}$ itself, through the expected log-likelihood. This is especially perceptible when $p_\theta(\boldsymbol{x} \mid \boldsymbol{z})$ is a Gaussian as in Equation (2.19), since increasing the reconstruction term then amounts to decreasing the MSE between the data points and their reconstructions.

Therefore, besides being a generative model, VAEs are also a popular basis for representation learning, more particularly in a disentangled latent space (Locatello et al., 2019; E. Mathieu et al., 2019) – partly thanks to the KLD term in Equation (2.22)

that regularizes the autoencoding part of the model. The interesting properties of learned representations through a VAE make them fit to better understand the data and control the generation process through the disentangled factors of variation, as pointed out by Kingma and Welling (2019).

VAEs also come with disadvantages and learning challenges, as also noted by the latter authors. Namely, they require a careful and more involved design than presented above to face recurring issues. For example, the blurriness of their outputs in the context of image generation has penalized VAEs in comparison to other generative models. Furthermore, they can be subject to optimization issues like posterior collapse occurring when $q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \approx p_{\boldsymbol{z}}$, which is a difficult state to escape once attained. Numerous works have analyzed these problems and proposed solutions, such as Alemi et al. (2018), Tomczak and Welling (2018), Dai and Wipf (2019), Loaiza-Ganem and Cunningham (2019), and Vahdat and Kautz (2020), including theoretical analyses, optimization improvements and sophistication of the considered distributions for priors and posteriors.

While some of these solutions have been considered in related works of our contributions, we do not employ them in this document. We only leverage the now standard β-VAE (Higgins, Matthey, et al., 2017) that modifies the ELBO objective by multiplying the KLD term of Equation (2.22) with a constant hyperparameter β:

$$\mathcal{L}^{(\beta)}_{\text{ELBO}}(\theta, \phi) \triangleq \mathbb{E}_{\boldsymbol{z} \sim q_\phi(\boldsymbol{z} \mid \boldsymbol{x})} \log p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) - \beta \cdot D_{\text{KL}}\big(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \,\big\|\, p(\boldsymbol{z})\big). \tag{2.25}$$

Note that this remains a lower bound of the log-likelihood if $\beta \geq 1$. β-VAEs have been shown to improve the quality and disentanglement of learned representations as well as the flexibility of VAEs (Alemi et al., 2018), especially for image generation.

2.3.1.2. Generative Adversarial Networks

GANs, introduced by Goodfellow, Pouget-Abadie, et al. (2014), take the opposite view of VAEs: they cannot optimize the log-likelihood of the data, even indirectly like VAEs with the ELBO. Instead, they rely on adversarial training to optimize the generator, which we assume to have the form of Equation (2.20). In this regard, the optimization objective of the generator is defined with respect to a discriminator, whose own objective is adverse to the generator's one.

Indeed, the goal of the discriminator f (with scalar outputs) is to distinguish between samples from the generated and target distribution. In the original formulation of Goodfellow, Pouget-Abadie, et al. (2014), f accordingly solves the following maximization problem, as a binary classifier:

$$\begin{aligned}
&\max_{f \in \mathcal{F}} \; \mathbb{E}_{\boldsymbol{x} \sim p_\theta}\Big[\log\big(1 - (\sigma \circ f)(\boldsymbol{x})\big)\Big] + \mathbb{E}_{\boldsymbol{y} \sim p_{\text{data}}}\big[\log(\sigma \circ f)(\boldsymbol{y})\big] \\
&\quad = \max_{f \in \mathcal{F}} \; \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}}\Big[\log\Big(1 - (\sigma \circ f)\big(g_\theta(\boldsymbol{z})\big)\Big)\Big] + \mathbb{E}_{\boldsymbol{y} \sim p_{\text{data}}}\big[\log(\sigma \circ f)(\boldsymbol{y})\big] \\
&\quad = \max_{f \in \mathcal{F}} \; \mathcal{L}_{\text{GAN}}(f, g_\theta),
\end{aligned} \tag{2.26}$$

with:

$$\mathcal{L}_{\text{GAN}}(f, g) \triangleq \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}\Big[\log\Big(1 - (\sigma \circ f)\big(g(\boldsymbol{z})\big)\Big)\Big] + \mathbb{E}_{\boldsymbol{y}\sim p_{\text{data}}}\big[\log(\sigma \circ f)(\boldsymbol{y})\big], \tag{2.27}$$

where σ is the sigmoid function and $\mathcal{F}$ the family of investigated discriminators. In turn, the generator adopts the opposite objective, yielding the following minimax optimization problem:

$$\min_{g\in\mathcal{G}} \max_{f\in\mathcal{F}} \; \mathcal{L}_{\text{GAN}}(f, g), \tag{2.28}$$

where $\mathcal{G}$ is the family of generators considered during training.

In practice, both $\mathcal{F}$ and $\mathcal{G}$ are families of neural networks respectively parameterized by ϑ and θ. To keep the cost of computing the discriminator reasonable and to avoid sigmoid saturation and gradient vanishing, the actual optimization of f_ϑ and g_θ differs from the minimax problem of Equation (2.28). Instead of computing a different discriminator from scratch for each generator g_θ, the vast majority of GANs jointly optimize a single f_ϑ together with g_θ in an alternating fashion. In other words, an optimization step of a GAN model, repeated until convergence, consists of the following:

- perform a few gradient ascent steps on f_ϑ with objective $\mathcal{L}^{\text{discr}}(f_\vartheta, g_\theta)$, with g_θ frozen;
- perform a gradient descent step on g_θ with objective $\mathcal{L}^{\text{gen}}(f_\vartheta, g_\theta)$, with f_ϑ frozen.

In the initial formulation of Goodfellow, Pouget-Abadie, et al. (2014), $\mathcal{L}^{\text{discr}} = \mathcal{L}^{\text{gen}} = \mathcal{L}_{\text{GAN}}$. Nevertheless, since the second term of $\mathcal{L}_{\text{GAN}}$ in Equation (2.27) does not depend on g_θ, this also amounts to having:

$$\mathcal{L}^{\text{discr}} = \mathcal{L}_{\text{GAN}}, \qquad \mathcal{L}^{\text{gen}}(f, g) = \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}\Big[\log\Big(1 - (\sigma \circ f)\big(g(\boldsymbol{z})\big)\Big)\Big]. \tag{2.29}$$

This initial formulation was originally justified by observing that, in the minimax expression of Equation (2.28), an optimal discriminator with infinite capacity would make the generator's loss to minimize equal to the Jensen-Shannon divergence between p_θ and p_{data}. However, GANs are notoriously hard to train: their optimization is challenging, they necessitate extensive hyperparameter tuning, and their generation results suffer from issues such as mode collapse (Goodfellow, 2016; M.-Y. Liu et al., 2021). This has led the community to propose a plethora of variants (Zhengwei Wang, She, and Ward, 2021).

Among them, Arjovsky, Chintala, and Bottou (2017) point out possible gradient issues with the hereinabove model and consequently proposed Wasserstein GANs (WGANs), provably providing the discriminator with good gradients by linking it to the optimal critic of the dual representation of the $\mathcal{W}_1$ Wasserstein distance. Corresponding objectives, as popularized by Gulrajani et al. (2017), are defined as:

$$\mathcal{L}_{\text{IPM}}(f, g) = \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}\Big[f\big(g_\theta(\boldsymbol{z})\big)\Big] - \mathbb{E}_{y\sim p_{\text{data}}}\big[f(\boldsymbol{y})\big], \tag{2.30}$$

$$\mathcal{L}^{\text{gen}}(f, g) = \mathcal{L}_{\text{IPM}}(f, g), \text{ or equivalently } \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}\Big[f\big(g_\theta(\boldsymbol{z})\big)\Big], \tag{2.31}$$

$$\mathcal{L}^{\text{discr}}(f, g) = \mathcal{L}_{\text{IPM}}(f, g) + \text{GP}_g(f), \tag{2.32}$$

where $\mathcal{L}_{\text{IPM}}(f, g)$ is the Integral Probability Metric (IPM) formula (Müller, 1997), and $\text{GP}_g(f)$ is a gradient penalty on f constraining it to be close to 1-Lipschitz over the support of p_{data} and p_θ.

Inspired gradient constraints, together with architectural advances and model changes, have led to high-performing models scaling to complex and high-dimensional image datasets (Brock, Donahue, and Simonyan, 2019; Karras, Laine, Aittala, et al., 2020). As things stand, even though they may lack latent space manipulation properties of e.g. VAEs, GANs constitute the most successful and popular generative model when it comes to synthesize realistic images.

2.3.1.3. Other Categories

Besides VAEs and GANs, there are multiple other types of deep generative models in the literature, like deep Boltzmann machines (Salakhutdinov and Hinton, 2009) and score-based models (Y. Song et al., 2021). We succinctly describe hereinbelow two likelihood-based models that are relevant in our setting as they have been used to handle sequences – cf. Section 2.1.

Autoregressive models. Such models rely on the assumption that the data is structured, i.e. a data point x can be decomposed in components $\boldsymbol{x} = (x_1, \dots, x_c)$. They consequently factorize their likelihood in an autoregressive manner as:

$$p_\theta(\boldsymbol{x}) = \prod_{i=1}^{c} p_\theta(x_i \mid x_1, \dots, x_{i-1}) \quad \Leftrightarrow \quad \log p_\theta(\boldsymbol{x}) = \sum_{i=1}^{c} \log p_\theta(x_i \mid x_1, \dots, x_{i-1}), \tag{2.33}$$

and explicitly compute each term of the latter sum by choosing as $p_\theta(x_i \mid x_1, \dots, x_{i-1})$ a simple distribution whose parameters are computed from $(x_1, \dots, x_{i-1})$ via a neural network parameterized by θ. This technique is especially convenient for data with discrete values because the previous conditional probabilities become, without loss of generality, categorical distributions. This explains their prominent use in NLP for e.g. language modeling (Devlin et al., 2019). They have, nevertheless, also found applications to other data types like for images, by considering their pixel values to belong to their original integer range $[\![0, 255]\!]$ (Oord, Kalchbrenner, and Kavukcuoglu, 2016).

Autoregressive generative models, by nature, require to model sequences of data components, thereby demanding complex dedicated architectures (Larochelle and Murray, 2011; van den Oord, Kalchbrenner, et al., 2016; Vaswani et al., 2017) which can induce a high computational cost. These architectures generally involve standard sequential architectures, which we describe in Section 2.1.

Normalizing flows. Methods based on normalizing flows (Kobyzev, Prince, and Brubaker, 2020) choose the setting of Equation (2.20) but additionally ensure that the generator is invertible, inducing an analytical expression of the model log-likelihood:

$$\log p_\theta(\boldsymbol{x}) = \log p_{\boldsymbol{z}}(\boldsymbol{z}) - \log \det \mathrm{J}_{g_\theta}(\boldsymbol{z}), \qquad z = g_\theta^{-1}(\boldsymbol{x}), \tag{2.34}$$

where J_{g_θ} denotes the Jacobian matrix of g_θ and $\det \mathrm{J}_{g_\theta}(\boldsymbol{z})$ its determinant. Note that the data and latent dimensions should be equal in this case.

The main challenge of these methods deals with how to efficiently compute this Jacobian while ensuring that the generator is sufficiently powerful for the considered task. As for autoregressive models, this implies constraining the generator for these techniques to be applicable. Propositions in this regard facilitate the computation of the Jacobian determinant by specifically adapting the structure of the Jacobian matrix (Dinh, Sohl-Dickstein, and S. Bengio, 2017; Kingma and Dhariwal, 2018) to be e.g. triangular.

On the other hand, other invertible models circumvent this analytic computation thanks to numerical approximation methods, with for instance the works of Behrmann et al. (2019) and Grathwohl et al. (2019). Both leverage the ODE connection with neural networks established in Section 2.1. On the one hand, Grathwohl et al. (2019) propose as invertible transformation an ODE parameterized by neural networks whose invertibility is ensured by the Cauchy-Lipschitz theorem, with a likelihood computed by integrating another ODE. On the other hand, Behrmann et al. (2019) remain in discrete-time by using as invertible transformation a bijective residual network, obtained by constraining the Lipschitzness of the residuals. This prevents the model from being hindered by numerical errors in ODE integration which could invalidate the invertibility of the discretized ODE.

2.3.2. Sequential Deep Generative Models

The generative models presented until now in the context of static data have also been used for sequential data. The general formulation that we adopt for their presentation is directly applicable to such temporal data. However, the nature and complexity of the latter call for specifically designed models.

For example, a desirable property of temporal generative models would be to generate longer sequences than those they have been trained on, which is only possible when specific architectures are used. There are abundant neural network architectures that are specially adapted for sequential data, mostly based on RNNs or any other sequential architecture of Section 2.1. They can be used as a direct replacement for the generator g_θ or any network involved during the training of the latter. We more specifically discuss these architectures in Section 2.1 and instead focus in the rest of this discussion on specific generative modeling advances towards better handling time series.

Their breadth of application being immense with multiple sources and types of temporal data, a thorough review of generative models in this setting would be outside of the scope of this document. We instead opt for highlighting general research axes in the literature of sequential generative modeling, listed in the following.

2.3.2.1. Temporally Aware Training Objectives

One of the manners to adapt existing methods such as those of Section 2.3.1 is to further specialize their training objective to take into account the temporality of the data, without necessarily changing the structure of the generative model. This is

beneficial because objectives tailored for static data can bias generative models towards producing undesirable effects (M. Mathieu, Couprie, and LeCun, 2016; Le Guen and Thome, 2019), such as blurry outputs for videos.

Proposed adaptations of training objectives can be generally applicable to any kind of time series by nature. For instance, T. Xu et al. (2020) adapt the WGAN objective by replacing the optimal transport view of $\mathcal{W}_1$ with a causal optimal transport paradigm, thus constraining the adversarial objective to account for the causality brought by the temporality of the data. Nonetheless, proposed methods are most often specifically tailored for the considered data, such as videos (T.-C. Wang et al., 2018), audio (Dhariwal et al., 2020) and low-dimensional data (Cuturi and Blondel, 2017). For example, the general GAN discriminator is replaced by T.-C. Wang et al. (2018) with two discriminators of different architectures and nature: a first one acting on video frames only to assess their individual quality and a second one taking as input the whole video to consider the temporal consistency of the produced sequence.

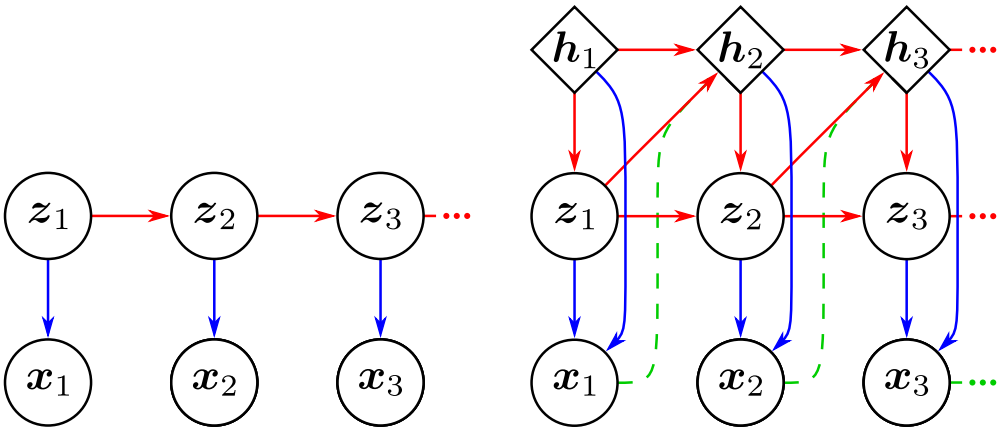
While this research direction is promising, we rather deal in this thesis with structural changes – i.e., modifications of p_θ – to obtain temporal generative models, which we discuss in the rest of this section.

2.3.2.2. Stochastic and Deterministic Models for Sequence-to-Sequence Tasks

A standard extension of generative models presented earlier is to tackle conditional generation problems (Mirza and Osindero, 2014; Sohn, H. Lee, and X. Yan, 2015), where the goal is to generate data points $\boldsymbol{x}$ under some condition $\boldsymbol{c}$, i.e. the generative model instead corresponds to the conditional probability $p_\theta(\boldsymbol{x} \mid \boldsymbol{c})$ trained to imitate $p_{\text{data}}(\boldsymbol{x} \mid \boldsymbol{c})$. For static data, a typical example is class-conditional generation, e.g. synthesizing an image of a given object (Odena, Olah, and Shlens, 2017).

For sequential data, conditional generation is also applied to sequence-to-sequence tasks, for which an input sequence conditions the output sequence. This includes sequence transformations (van den Oord, Dieleman, et al., 2016; T.-C. Wang et al., 2018) as well as prediction tasks, consisting in forecasting the next future time steps of a series based on some previous conditioning steps.

Sequence conditioning may be strong enough to fully or almost completely determine the corresponding output for some data types because conditioning steps contain decisive information about the dynamics of the observed series. In this case, the true conditional $p_{\text{data}}(\boldsymbol{x} \mid \boldsymbol{c})$ is a Dirac, or close to a Dirac. This happens, for instance, in fully observable physical phenomena driven by ODEs, where the latter ensure that sufficient observations can determine the whole process, or in videos where visual features and movements can be predictable in the short term. This has led authors in fields where this observation stands to choose, often implicitly, a Dirac as $p_\theta(\boldsymbol{x} \mid \boldsymbol{c})$ that is centered at the point outputted by the generator. In this setting, the generator becomes a simple deterministic regressor, trained to predict a function of its inputs. While this can be achieved via usual loss functions like the MSE, some peculiar techniques of generative modeling can also be applied in this case to improve the prediction quality, such as adversarial losses (M. Mathieu, Couprie, and LeCun, 2016; Vondrick and Torralba, 2017).



(a) Example of a state-space model: a deep Markov model (Krishnan, Shalit, and D. Sontag, 2017).

(b) VRNN model (Chung, Kastner, et al., 2015). Green dashed arrows highlight the observation-autoregressive feedback loop.

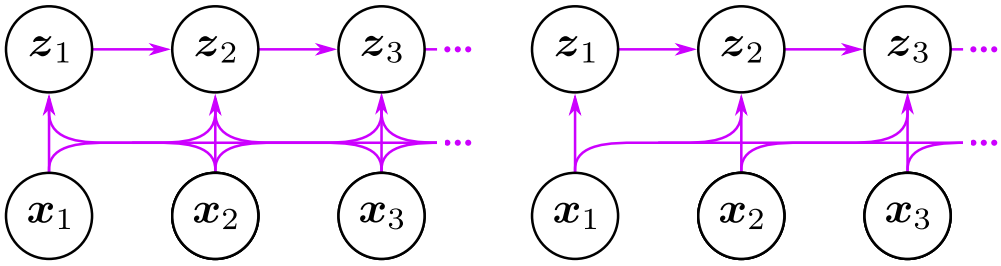
Figure 2.3.: Graphical representations of the factorization of p_θ for two standard sequential latent variable models. Diamonds and circles represent, respectively, deterministic and stochastic states, conditionally to parent states.

This determinacy assumption simplifies learning, but it may also be a crude approximation of the observed phenomena. Indeed, real-world data most often present some form of stochasticity, whether from observation noise or because the underlying phenomenon includes unpredictable events. Thus, a longstanding research direction for sequence modeling is to design generative methods that can take into account the stochasticity of the data (Schuster, 2000; Graves, 2013; Babaeizadeh et al., 2018; J. Jia and Benson, 2019). This may necessitate a more involved model design to adapt VAEs and GANs, for instance, to stochastic temporal data. We address this related work in the following.

2.3.2.3. Latent Generative Temporal Structure

Apart from autoregressive models which inherently present a temporal structure as noted in Section 2.3.1.3, the specification of generative models relying on transforming latent variables like in Equations (2.19) and (2.20) is not sequential. The main response to this issue lies in sequential latent variable models, which we concisely summarize here; we refer to the work of Fraccaro (2018) for a detailed exposition. Most of these methods relying on deep neural networks are trained via variational inference within the VAE framework.

In the static case, the latent variable $\boldsymbol{z}$, instead of being Gaussian, can be decomposed in a hierarchy of $\boldsymbol{z}_1$, $\boldsymbol{z}_2$, etc., with learned conditional non-linear priors $p_\theta(\boldsymbol{z}_{i+1} \mid \boldsymbol{z}_i)$ whose factorization yields a particular development of the ELBO – see e.g. the work of



(a) Smoothing: the whole input sequence is used for inference.

(b) Filtering: only the present and the past steps are used for inference.

Figure 2.4.: Graphical representations of two possible factorizations of q_ϕ for the deep Markov model of Figure 2.3(a).

C. K. Sønderby, Raiko, et al. (2016). Similarly, such a hierarchy of latent variables may be leveraged in a temporal fashion.

A representative example is deep Markov models (Krishnan, Shalit, and D. Sontag, 2017), whose factorization of p_θ is illustrated in Figure 2.3(a) and described as follows:

$$p_\theta(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{z}_1) \prod_{t=2}^{T} p_\theta(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1}) \prod_{t=1}^{T} p_\theta(\boldsymbol{x}_t \mid \boldsymbol{z}_t), \tag{2.35}$$

where T is the length of $\boldsymbol{x}$. As a consequence of its temporal nature, an advantage of this type of model is that it applies to any sequence length T by relying on its transition – $p_\theta(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1})$ – and emission – $p_\theta(\boldsymbol{x}_t \mid \boldsymbol{z}_t)$ – processes. To efficiently train such a model, the recognition network q_ϕ (see Section 2.3.1.1) is designed to mirror the factorization of p_θ, with for example, following Krishnan, Shalit, and D. Sontag (2017) and as illustrated in Figure 2.4(a):

$$q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) = q_\phi(\boldsymbol{z}_1 \mid \boldsymbol{x}) \prod_{t=2}^{T} q_\phi(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1}, \boldsymbol{x}), \tag{2.36}$$

yielding a specific development of the ELBO of Equation (2.22):

$$\begin{aligned} \log p_\theta(\boldsymbol{x}) \geq \mathbb{E}_{\boldsymbol{z} \sim q_\phi(\boldsymbol{z} \mid \boldsymbol{x})} \sum_{t=1}^{T} \log p_\theta(\boldsymbol{x}_t \mid \boldsymbol{z}_t) - D_{\mathrm{KL}}\big(q_\phi(\boldsymbol{z}_1 \mid \boldsymbol{x}) \,\big\|\, p(\boldsymbol{z}_1)\big) \\ - \mathbb{E}_{\boldsymbol{z} \sim q_\phi(\boldsymbol{z} \mid \boldsymbol{x})} \sum_{t=2}^{T} D_{\mathrm{KL}}\big(q_\phi(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1}, \boldsymbol{x}) \,\big\|\, p_\theta(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1})\big). \end{aligned} \tag{2.37}$$

Note that in the latter equation, $q_\phi(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1}, \boldsymbol{x})$ makes the inference of $\boldsymbol{z}_t$ depend on the whole sequence $\boldsymbol{x}$, which is called smoothing. In another setting called filtering that we adopt in Chapter 4, one can choose to make it only depend on the past and

present steps $\boldsymbol{x}_{1:t}$ as illustrated in Figure 2.4(b), i.e.:

$$q_\phi(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1}, \boldsymbol{x}) = q_\phi(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1}, \boldsymbol{x}_{1:t}), \tag{2.38}$$

where $\boldsymbol{x}_{t:t'}$ corresponds to the following subsequence of $\boldsymbol{x}$:

$$\boldsymbol{x}_{t:t'} = (\boldsymbol{x}_t, \boldsymbol{x}_{t+1}, \ldots, \boldsymbol{x}_{t'}). \tag{2.39}$$

While smoothing should yield a better likelihood thanks to the received information via q_ϕ from both the past and future time steps, filtering endows the model with increased flexibility, as the latter can be employed in an online setting where the future is not available.

The previously described model belongs to the family of state-space models, where each observation $\boldsymbol{x}_t$ is decoded from a single state $\boldsymbol{z}_t$ independently of the other observations; more formally, the state-space hypothesis requires the likelihood p_θ to admit a factorization of the form of Equation (2.35). This type of generative structure has received substantial attention because it implies that the latent state of each time step $\boldsymbol{z}_t$ contains the information necessary to represent and decode $\boldsymbol{x}_t$ as well as forecast $\boldsymbol{x}_{>t}$ (Karl et al., 2017), without relying on past or future states. Such representation learning paves the way for interesting downstream applications leveraging these complete representations of a system, like in reinforcement learning (Gregor et al., 2019).

Another line of research investigates, in opposition to state-space modeling, what we call in this document observation-autoregressive models, where the past predictions $\boldsymbol{x}_{1:t}$ are fed back in the state temporal model to produce the future state $\boldsymbol{z}_{t+1}$ and subsequently its corresponding next prediction $\boldsymbol{x}_{t+1}$. It has become especially popular since the work of Bayer and Osendorfer (2014) that initiated the introduction of RNNs in sequential latent variable models. The observation-autoregressivity then arises from the hidden state of the RNN which takes as inputs the observations and is used to produce the latent states $\boldsymbol{z}$. A popular model of this kind is the Variational Recurrent Neural Network (VRNN) of Chung, Kastner, et al. (2015), whose generative process is specified in Figure 2.3(b). It features a deterministic hidden state $\boldsymbol{h}_{t+1}$ taking as input $\boldsymbol{h}_t$, $\boldsymbol{z}_t$ as well as $\boldsymbol{x}_t$ in an autoregressive manner to produce the next state $\boldsymbol{z}_{t+1}$ and observation $\boldsymbol{x}_{t+1}$. This observation-autoregressive feedback loop facilitates learning by inputting real observations during training.

We point out that the presented latent variables models historically originate from Kalman filters (Kalman, 1960), whose transition function between $\boldsymbol{z}$ variables is linear and Gaussian. This enables as a consequence exact inference contrary to the deep Markov model of Figure 2.3(a), besides offering missing input computation abilities. Despite their simplicity, Kalman filters remain relevant within the field of deep learning when coupled with other generative techniques (Krishnan, Shalit, and D. Sontag, 2015). Among its numerous other applications, its analytically computable likelihood allows authors to employ it in a latent space by combining it with likelihood-based models such as VAEs (Fraccaro, Kamronn, et al., 2017) and normalizing flows (de Bézenac, Rangapuram, et al., 2020).

Part II.

Time Series Representation Learning

We present in the upcoming chapter our first contribution, which deals with unsupervised representation learning for time series, and led to the following publication in an international conference:

> Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi (2019). "Unsupervised Scalable Representation Learning for Multivariate Time Series". In: *Advances in Neural Information Processing Systems*. Ed. by Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily Fox, and Roman Garnett. Vol. 32. Curran Associates, Inc., pp. 4650–4661.

Chapter 3.

Unsupervised Scalable Representation Learning for Time Series

3.1. Introduction

We investigate in this chapter the topic of unsupervised general-purpose representation learning for time series. In spite of the increasing amount of work about representation learning in fields like natural language processing (Young et al., 2018) or videos (Denton and Birodkar, 2017), as described in Section 2.2, few articles explicitly deal with unsupervised representation learning for time series without structural assumption on non-temporal data.

This problem is indeed challenging for various reasons. First, real-life time series are rarely or sparsely labeled. Therefore, unsupervised representation learning would be strongly preferred. Second, methods need to deliver compatible representations while allowing the input time series to have unequal lengths. Third, scalability and efficiency both at training and inference time are crucial, in the sense that the techniques must work for both short and long time series encountered in practice.

Hence, we propose in the following an unsupervised method to learn general-purpose representations for multivariate time series that comply with the issues of varying and potentially high lengths of the studied time series. To this end, we introduce a novel unsupervised loss training a scalable encoder, shaped as a deep convolutional neural network with dilated convolutions (van den Oord, Dieleman, et al., 2016) and outputting fixed-length vector representations regardless of the length of its input. This loss is built as a triplet loss employing time-based negative sampling, taking advantage of the encoder resilience to time series of unequal lengths. To our knowledge, it is the first fully unsupervised triplet loss in the literature of time series.

We assess the quality of the learned representations on various datasets to ensure their universality. In particular, we test how our representations can be used for classification tasks on the standard datasets in the time series literature, compiled in the UCR repository (Dau et al., 2018). We show that our representations are general and transferable, and that our method outperforms concurrent unsupervised methods and even matches the state of the art of non-ensemble supervised classification techniques. Moreover, since UCR time series are exclusively univariate and mostly short, we also evaluate our representations on the recent UEA multivariate time series repository (Bagnall, Dau, et al., 2018), as well as on a real-life dataset including very

long time series, on which we demonstrate scalability, performance and generalization ability across different tasks beyond classification.

This chapter is organized as follows. Section 3.2 specifies the context of this work beyond Chapter 2 with respect to unsupervised representation learning and triplet losses for time series in the literature. Section 3.3 describes the unsupervised training of the encoder, while Section 3.4 details the architecture of the latter. Section 3.5 then provides results of the experiments that we conducted to evaluate our method, while Section 3.6 more lightly discusses some details of our approach. Finally, the supplementary material referenced in this chapter is available in Appendix A.

3.2. Related Work

Unsupervised learning for time series. To our knowledge, apart from those dealing with videos or high-dimensional data (Srivastava, Mansimov, and Salakhudinov, 2015; Denton and Birodkar, 2017; Villegas, J. Yang, Hong, et al., 2017; van den Oord, Yazhe Li, and Vinyals, 2018), few recent works tackle unsupervised representation learning for time series, especially at the time this work was developed. Fortuin et al. (2019) deal with a related but different problem to this work, by learning temporal representations of time series that represent well their evolution. Hyvarinen and Morioka (2016) learn representations on evenly sized subdivisions of time series by learning to discriminate between those subdivisions from these representations. Lei et al. (2019) expose an unsupervised method designed so that the distances between learned representations mimic the standard distance Dynamic Time Warping (DTW) between time series. Malhotra et al. (2017) design an encoder as a recurrent neural network, jointly trained with a decoder as a sequence-to-sequence model to reconstruct the input time series from its learned representation. Finally, L. Wu et al. (2018) compute feature embeddings generated in the approximation of a carefully designed and efficient kernel.

However, these methods either are neither scalable nor suited to long time series (due to the sequential nature of a recurrent network, or to the use of DTW with a quadratic complexity with respect to the input length), are tested on no or few standard datasets and with no publicly available code, or do not provide sufficient comparison to assess the quality of the learned representations. Our scalable model and extensive analysis aim at overcoming these issues, besides outperforming these methods.

Triplet losses. Triplet losses have recently been widely used in various forms for representation learning in different domains (Mikolov, Sutskever, et al., 2013; Schroff, Kalenichenko, and Philbin, 2015; L. Y. Wu et al., 2018) and have also been theoretically studied (Saunshi et al., 2019). However, they have not found much use for time series apart from audio (Bredin, 2017; R. Lu et al., 2017; Jansen et al., 2018), and rarely, to our knowledge, in a fully unsupervised setting, as existing works assume the existence of class labels or annotations in the training data.

Closer to our work even though focusing on a different, more specific task, Jansen et al. (2018) and Turpault, Serizel, and E. Vincent (2019) learn audio embeddings respectively in an unsupervised and semi-supervised setting. In particular, they circumvent the

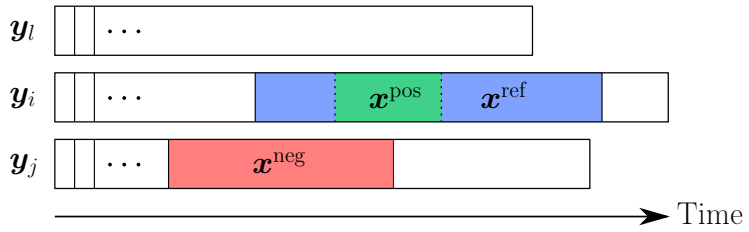


Figure 3.1.: Choices of x^{ref}, x^{pos} and x^{neg}, respectively reference, positive and negative examples, in the introduced triplet loss.

absence of labels with slight modifications of reference data points x^{ref} with audio-adapted transformations to create close audio clips x^{pos}, providing a label-free method to mine positive example pairs. Additionally, Logeswaran and H. Lee (2018) train a sentence encoder to recognize, among randomly chosen sentences, the true context of another sentence, which is a difficult method to adapt to time series.

Our method instead relies on a more general choice of positive samples, learning similarities using subsampling. This resembles, in the context of time series, other unsupervised learning works on images leveraging random crops to create positive samples (T. Wang and Isola, 2020).

3.3. Unsupervised Training

We seek to train an encoder-only architecture, avoiding the need to jointly train it with a decoder like in autoencoder-based standard representation learning methods as done by Malhotra et al. (2017). To this end, we introduce a novel triplet loss for time series, inspired by the seminal word representation learning method known as word2vec (Mikolov, Sutskever, et al., 2013). The proposed triplet loss uses original time-based sampling strategies to overcome the challenge of learning on unlabeled data. As far as we know, this work is the first in the time series literature to rely on a triplet loss in a fully unsupervised setting.

The objective is to ensure that similar time series obtain similar representations, with no supervision to learn such similarity. Triplet losses help to achieve the former (Schroff, Kalenichenko, and Philbin, 2015), but require to provide pairs of similar inputs, thus challenging the latter. While previous supervised works for time series using triplet losses assume that data is annotated, we introduce an unsupervised time-based criterion to select pairs of similar time series taking into account time series of varying lengths, by following word2vec's intuition. The assumption made in the CBOW model of word2vec is twofold. The representation of the context of a word should probably be, on the one hand, close to the one of this word (Goldberg and Levy, 2014), and, on the other hand, distant from the ones of randomly chosen words, since they are probably unrelated to the original word's context. The corresponding loss then pushes pairs of (context, word) and (context, random word) to be linearly separable. This is

Algorithm 1: Choices of $\boldsymbol{x}^{\mathrm{ref}}$, $\boldsymbol{x}^{\mathrm{pos}}$ and $\left(\boldsymbol{x}_k^{\mathrm{neg}}\right)_{k\in[\![1,K]\!]}$ for an epoch over the dataset $(\boldsymbol{y}_i)_{i\in[\![1,N]\!]}$. $\mathrm{len}(x)$ denotes the length of the time series $\boldsymbol{x}$.

for $i \in [\![1, N]\!]$ **with** $l_i = \mathrm{len}(\boldsymbol{y}_i)$ **do**
  `// Choosing the length of positive and reference points`
  sample $l^{\mathrm{pos}} = \mathrm{len}(\boldsymbol{x}^{\mathrm{pos}})$ in $[\![1, s_i]\!]$ uniformly at random;
  choose $l^{\mathrm{ref}} = \mathrm{len}\left(\boldsymbol{x}^{\mathrm{ref}}\right)$ in $[\![l^{\mathrm{pos}}, l_i]\!]$ uniformly at random;
  `// Choosing positive and anchor points`
  sample $\boldsymbol{x}^{\mathrm{ref}}$ uniformly at random among subseries of $\boldsymbol{y}_i$ of length l^{ref};
  sample $\boldsymbol{x}^{\mathrm{pos}}$ uniformly at random among subseries of $\boldsymbol{x}^{\mathrm{ref}}$ of length l^{pos};
  `// Negative Sampling: choosing random subseries in the dataset`
  for $k \in [\![1, K]\!]$ **do**
    sample $j_k \in [\![1, N]\!]$ uniformly at random;
    sample $l_k^{\mathrm{neg}} = \mathrm{len}\left(\boldsymbol{x}_k^{\mathrm{neg}}\right)$ in $[\![1, \mathrm{len}\left(\boldsymbol{y}_{j_k}\right)]\!]$ uniformly at random;
    sample $\boldsymbol{x}_k^{\mathrm{neg}}$ uniformly at random among subseries of $\boldsymbol{y}_k$ of length l_k^{neg}.

called negative sampling.

We adapt this principle to time series as follows, and illustrate it in Figure 3.1. Let us consider a random subseries – i.e. a subsequence of a time series composed of consecutive time steps – $\boldsymbol{x}^{\mathrm{ref}}$ of a given time series $\boldsymbol{y}_i$ from the dataset. Then, on the one hand, the representation of $\boldsymbol{x}^{\mathrm{ref}}$ should be close to the one of any of its subseries $\boldsymbol{x}^{\mathrm{pos}}$ (a positive example). On the other hand, if we consider another subseries $\boldsymbol{x}^{\mathrm{neg}}$ (a negative example) chosen at random (in a different random time series $\boldsymbol{y}_j$ if several series are available, or in the same time series if it is long enough and not stationary), then its representation should be distant from the one of $\boldsymbol{x}^{\mathrm{ref}}$. Following the analogy with word2vec, $\boldsymbol{x}^{\mathrm{pos}}$ corresponds to a word, $\boldsymbol{x}^{\mathrm{ref}}$ to its context, and $\boldsymbol{x}^{\mathrm{neg}}$ to a random word. To improve the stability and convergence of the training procedure as well as the experimental results of our learned representations, we introduce, as in word2vec, several negative samples $\left(\boldsymbol{x}_k^{\mathrm{neg}}\right)_{k\in[\![1,K]\!]}$, chosen independently at random.

The objective corresponding to these choices to minimize during training can be thought of the one of word2vec with its shallow network replaced with a deep network f_θ with parameters θ, or formally:

$$-\log\sigma\left(f_\theta\left(\boldsymbol{x}^{\mathrm{ref}}\right)^\top f_\theta(\boldsymbol{x}^{\mathrm{pos}})\right) - \frac{1}{K}\sum_{k=1}^{K}\log\sigma\left(-f_\theta\left(\boldsymbol{x}^{\mathrm{ref}}\right)^\top f_\theta\left(\boldsymbol{x}_k^{\mathrm{neg}}\right)\right), \tag{3.1}$$

where σ is the sigmoid function. This loss pushes the computed representations to distinguish between $\boldsymbol{x}^{\mathrm{ref}}$ and $\boldsymbol{x}^{\mathrm{neg}}$, and to assimilate $\boldsymbol{x}^{\mathrm{ref}}$ and $\boldsymbol{x}^{\mathrm{pos}}$. Overall, the training procedure consists in traveling through the training dataset for several epochs (possibly using mini-batches), picking tuples $\left(\boldsymbol{x}^{\mathrm{ref}}, \boldsymbol{x}^{\mathrm{pos}}, \left(\boldsymbol{x}_k^{\mathrm{neg}}\right)_k\right)$ at random as detailed in Algorithm 1, and performing a minimization step on the corresponding loss for each pair, until training ends. The overall computational and memory cost is $\mathcal{O}\big(K \cdot c(f)\big)$,

where $c(f)$ is the cost of evaluating and backpropagating through f on a time series. Therefore, this unsupervised training is scalable as long as the encoder architecture is scalable as well.

We highlight that this time-based triplet loss leverages the ability of the chosen encoder to take as input time series of different lengths. By training the encoder on a range of input lengths going from one to the length of the longest time series in the train set, it becomes able to output meaningful and transferable representations regardless of the input length, as shown in Section 3.5. The length of the negative examples is chosen at random in Algorithm 1 for the most general case; however, their length can also be the same for all samples and equal to the length of $\boldsymbol{x}^{\text{pos}}$. The former case is used when time series in the dataset do not have the same lengths. The latter case is suitable when all time series in the dataset have equal lengths, as it speeds up the training procedure thanks to computation factorizations and does not empirically affect performances.

This training procedure is interesting in that it is efficient enough to be run over long time series (see Section 3.5) with a scalable encoder (see Section 3.4), thanks to its decoder-less design and the separability of the loss, on which a backpropagation per term can be performed to save memory. We use the latter optimization for multivariate time series of lengths larger than 10 000.

3.4. Encoder Architecture

We explain and present in this section our choice of architecture for the encoder, which is motivated by three requirements: it must extract relevant information from time series; it needs to be time- and memory-efficient, both for training and testing; and it has to allow variable-length inputs. We choose to use deep neural networks with exponentially dilated causal convolutions to handle time series. While they have been popularized in the context of sequence generation (van den Oord, Dieleman, et al., 2016), they have never been used for unsupervised time series representation learning. They offer several advantages.

Compared to RNNs, which are inherently designed for sequence-modeling tasks and thus sequential, these networks are scalable as they allow efficient parallelization on modern hardware such as GPUs. Besides this demonstrated efficiency, exponentially dilated convolutions have also been introduced to better capture, compared to full convolutions, long-range dependencies at constant depth by exponentially increasing the receptive field of the network (van den Oord, Dieleman, et al., 2016; F. Yu and Koltun, 2016; S. Bai, Kolter, and Koltun, 2018).

CNNs have also been demonstrated to be performant on various aspects for sequential data. For instance, while the issue of exploding and vanishing gradients of RNNs has received significant attention and workarounds, as mentioned in Section 2.1.1.2, recurrent networks are still outperformed by convolutional networks on this aspect (S. Bai, Kolter, and Koltun, 2018). On the specific domains of time series classification, which is an essential part of our experimental evaluation, and forecasting, deep neural networks have recently been successfully used as well (S. Bai, Kolter, and Koltun, 2018;

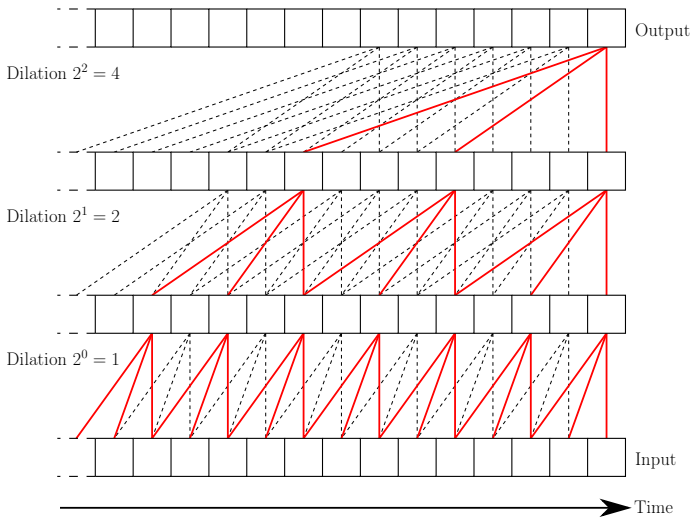


Figure 3.2.: Illustration of three stacked dilated causal convolutions. Lines between each sequence represent their computational graph. Red solid lines highlight the dependency graph for the computation of the last value of the output sequence, showing that no future value of the input time series is used to compute it.

Ismail Fawaz et al., 2019).

Our model is particularly based on stacks of dilated causal convolutions like in the model of van den Oord, Dieleman, et al. (2016) and as shown in Figure 3.2, which map a sequence to a sequence of the same length, such that the i-th element of the output sequence is computed using only values up until the i-th element of the input sequence, for all i. It is thus called causal, since the output value corresponding to a given time step is not computed using future input values. An interesting consequence of this choice is that causal convolutions alleviate the disadvantage of not using recurrent networks at testing time. Indeed, recurrent networks can be used in an online fashion, thus saving memory and computation time during testing. In our case, causal convolutions organize the computational graph so that, in order to update its output when an element is added at the end of the input time series, one only has to evaluate the highlighted graph shown in Figure 3.2 rather than the full graph.

Inspired by S. Bai, Kolter, and Koltun (2018), we build each layer of our network as a combination of causal convolutions, weight normalizations (Salimans and Kingma, 2016), leaky ReLUs and residual connections (see Figure 3.3(a)). Each of these layers is given an exponentially increasing dilation parameter (2^i for the i-th layer); in the example of Figure 3.2 with a convolutional kernel size of 3, the receptive field of an output of the resulting causal CNN with d layers would be of the order of 2^{d+1}. The output of this causal network is then given to a global max pooling layer squeezing the temporal dimension and aggregating all temporal information in a fixed-size vector (as

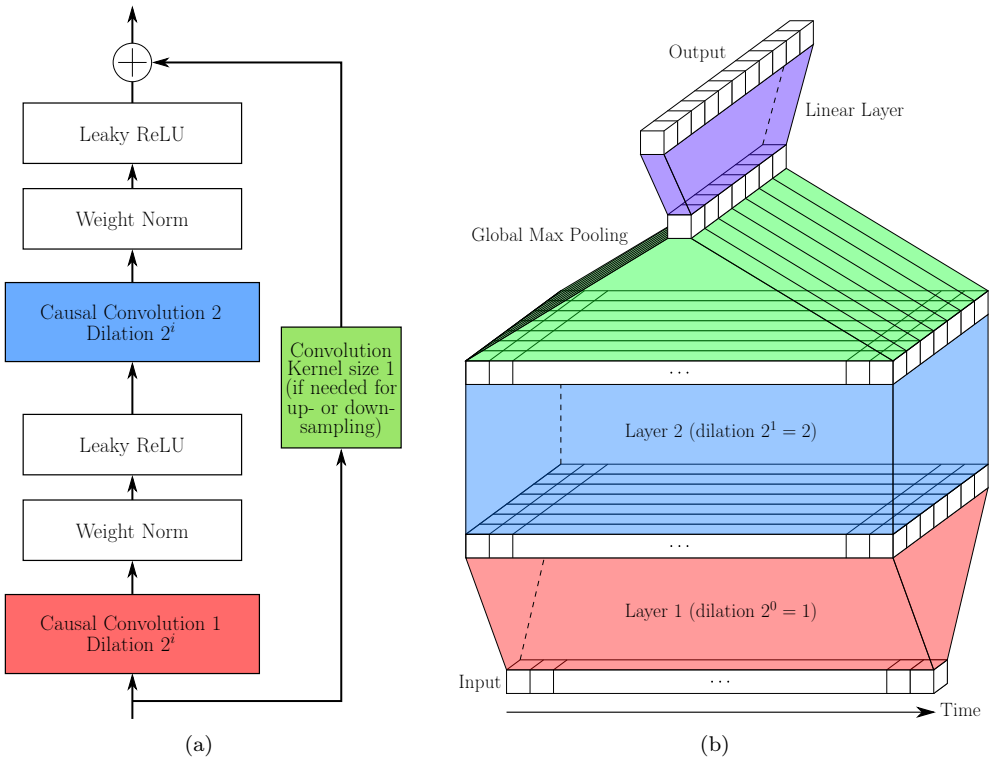


Figure 3.3.: (a) Composition of the i-th layer of the chosen architecture. (b) Example of the whole encoder architecture with two causal convolution layers.

proposed by Zhiguang Wang, Z. Yan, and Oates (2017) in a supervised setting with full convolutions). A linear transformation of this vector is then the output of the encoder, with a fixed size, that is independent of the input length. An illustration of this architecture is provided in Figure 3.3(b).

3.5. Experimental Results

We review in this section experiments conducted to investigate the relevance of the learned representations. The code corresponding to these experiments is publicly available.[1] The full training process and hyperparameter choices are detailed in Appendix A.1. We use Python 3.6.8 for the implementation, with PyTorch 0.4.1 (Paszke et al., 2019) for neural networks and scikit-learn (Pedregosa et al., 2011) for Support-Vector Machines (SVMs). Each encoder is trained using the Adam optimizer (Kingma and Ba, 2015) on a single Nvidia Titan Xp GPU with CUDA 9.0, unless stated otherwise.

Hyperparameter choices. Selecting hyperparameters for an unsupervised method is challenging. Indeed, it would be possible to choose them based on the performance of the representations in downstream tasks. However, the plurality of downstream tasks is usually supervised, hence this method of selection would indirectly introduce supervision in our encoder learning procedure. Therefore, like L. Wu et al. (2018), we choose for each considered dataset archive a single set of hyperparameters regardless of the downstream task. As a consequence, we highlight that we perform no hyperparameter optimization of the unsupervised encoder architecture and training parameters for any task, unlike other unsupervised works such as TimeNet (Malhotra et al., 2017).

Such a fixed set of hyperparameter values for all datasets especially impacts the choice of the depth of the causal CNN in the encoder, since it cannot be easily adapted to the length of the time series in the training dataset. To circumvent this problem, we opt for a causal CNN depth allowing its last output to have a receptive field of the order of 5000. This corresponds to a balance between the short time series of a hundred time steps and the long ones of up to dozens or hundreds of thousands steps.

3.5.1. Classification

We first assess the quality of our learned representations on supervised tasks in a standard manner (W. Xu, X. Liu, and Gong, 2003; Dosovitskiy, Springenberg, et al., 2014) by using them for time series classification. In this setting, we show that:

- our method outperforms state-of-the-art unsupervised methods, and notably achieves performance close to the supervised state of the art;
- strongly outperforms supervised deep learning methods when data is only sparsely labeled;

[1] https://github.com/White-Link/UnsupervisedScalableRepresentationLearningTimeSeries.

- produces transferable representations.

For each considered dataset with a train / test split, we unsupervisedly train an encoder using its training set. We then train an SVM with radial basis function kernel on top of the learned features using the training labels of the dataset and output the corresponding classification score on the testing set. As our training procedure encourages representations of different time series to be separable, observing the classification performance of a simple SVM on these features enables is to check their quality (L. Wu et al., 2018). Using SVMs also allows, when the encoder is trained, an efficient training procedure both in terms of time (training is a matter of minutes in most cases) and space.

As K has a significant impact on the performance, we present a combined version of our method, where representations computed by encoders trained with different values of K are concatenated (see Appendix A.1.3 for more details). This enables our learned representations with different parameters to complement each other and to remove some noise in the classification scores. The proposed combination also acts as an ensembling strategy, which is also adopted by most baselines considered in this experimental study.

3.5.1.1. Univariate Time Series

We present accuracy scores for all 128 datasets of the new iteration of the UCR archive (Dau et al., 2018), which is a standard set of varied univariate datasets. We report in Table 3.1 scores for only some UCR datasets, while scores for all datasets are reported in Appendix A.2.

We first compare our scores to the two concurrent methods of this work, TimeNet (Malhotra et al., 2017) and RWS (L. Wu et al., 2018), which are two unsupervised methods also training a simple classifier on top of the learned representations, and reporting their results on a few UCR datasets.

We also perform comparisons on the first 85 datasets of the archive[2] to the four best classifiers of the supervised state of the art studied by Bagnall, Lines, et al. (2017): COTE – replaced with its improved version HIVE-COTE (Lines, Taylor, and Bagnall, 2018) –, ST (Bostrom and Bagnall, 2015), BOSS (Schäfer, 2015) and EE (Lines and Bagnall, 2015). HIVE-COTE is a powerful ensemble method using many classifiers in a hierarchical voting structure; EE is a simpler ensemble method; ST is based on shapelets and BOSS is a dictionary-based classifier.[3] We add DTW – one-nearest-neighbor classifier with the standard distance DTW (Berndt and Clifford, 1994) as measure – as a baseline. HIVE-COTE includes ST, BOSS, EE and DTW in its ensemble, and is thus expected to outperform them.

[2] The new UCR archive includes 43 new datasets on which no reproducible results of state-of-the-art methods were produced at the time that this work was conducted. Still, we provide complete results for our method on these datasets in Appendix A.2, Table A.4, along with those of DTW, the only other method for which they are available.

[3] While ST and BOSS are also ensembles of classifiers, we choose not to qualify both of them as ensembles since their ensemble only includes variations of the same proposed classification method, like in our case with multiple values of K.

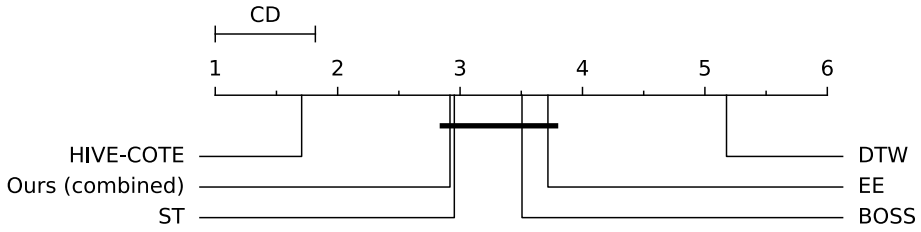


Figure 3.4.: Critical difference diagram of the average ranks of the compared classifiers for the Nemenyi test, obtained with Orange (Demšar et al., 2013).

Additionally, we compare our method to the ResNet method of Zhiguang Wang, Z. Yan, and Oates (2017), which is the best supervised neural network method studied in the review of Ismail Fawaz et al. (2019). We compile their results separately from the other baselines in the following because a fraction of the models produced by Ismail Fawaz et al. (2019) are trained on the old version of the archive, whereas our model and the other baselines are trained on its most recent release where some datasets are altered.

Performance. Comparison with the unsupervised state-of-the-art (with full results in Appendix A.2, Table A.3), indicates that our method consistently matches or outperforms both unsupervised methods TimeNet and RWS (on 11 out of 12 and 10 out of 11 UCR datasets), showing its performance. Unlike our work, code and full results on the UCR archive are not provided for these methods, hence the incomplete results.

When comparing to the supervised non-neural-network state of the art, we observe in Figures 3.4 and 3.5 that our method is globally the second-best one (with average rank 2.92), only beaten by HIVE-COTE (1.71) and equivalent to ST (2.95). Thus, our unsupervised method beats several recognized supervised classifier. It is only preceded by a powerful ensemble method; this is expected since the latter takes advantage of numerous classifiers and data representations. Additionally, Figure 3.6 shows that our method has the second-best median for the ratio of accuracy over maximum achieved accuracy, behind HIVE-COTE and above ST. We emphasize in particular that our method, by matching the performance of ST, is at the level of the best performing method included in HIVE-COTE. While it could be integrated to HIVE-COTE to improve the performance of the latter, this is beyond the scope of this work and requires significant technical work, as HIVE-COTE is implemented in Java and ours in Python.

Finally, results reported from the study of Ismail Fawaz et al. (2019) for the fully supervised ResNet (Appendix A.2,Table A.3) show that it expectedly outperforms our method on 63% out of the 71 UCR datasets for which their model is comparable to ours. Nonetheless, this natural discrepancy between our unsupervised method and this supervised model is remarkable as the latter outperforms ours only by a moderate

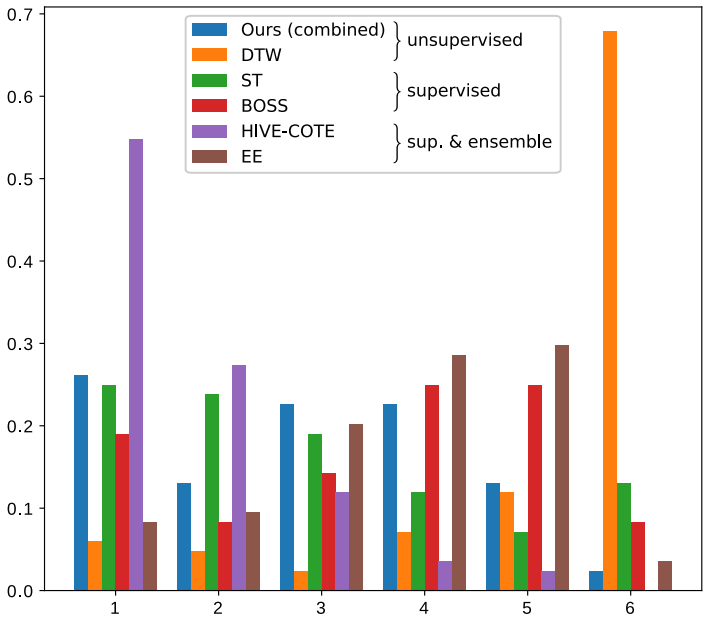


Figure 3.5.: Distribution of ranks of compared methods for the first 85 UCR datasets.

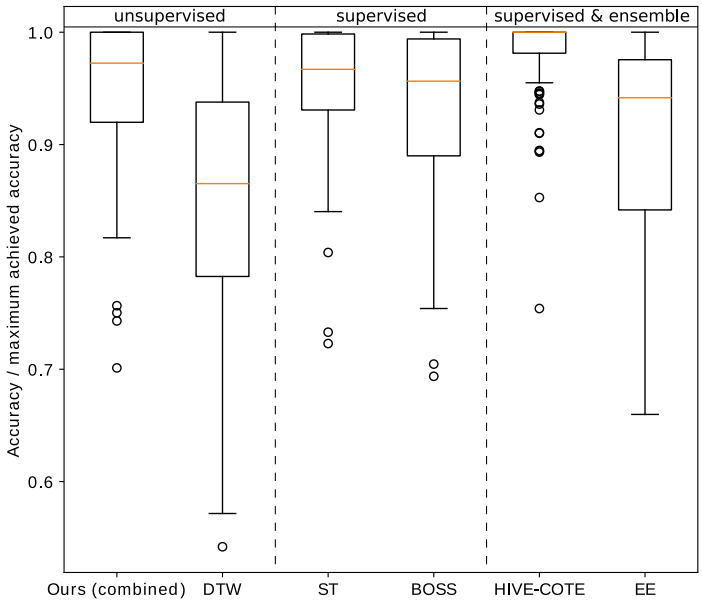


Figure 3.6.: Boxplot of the ratio of the accuracy versus maximum achieved accuracy (higher is better) for compared methods on the first 85 UCR datasets.

Table 3.1.: Accuracy scores of variants of our method compared with other supervised and unsupervised methods, on some UCR datasets. Results for the whole archive are available in Appendix A.2, Tables A.1, A.2 and A.4. Bold and underlined scores respectively indicate the best and second-best (when there is no tie for first place) performing methods.

Dataset	Unsupervised					Supervised			
	Ours				DTW	ST	BOSS	Ensemble	
	$K = 5$	$K = 10$	Combined	FordA				HIVE-COTE	EE
CricketX	0.700	0.713	0.777	0.682	0.754	0.772	0.736	**0.823**	<u>0.813</u>
DiatomSizeReduction	**0.993**	0.984	**0.993**	0.974	0.967	0.925	0.931	0.941	0.944
ECGFiveDays	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.984	**1.000**	**1.000**	0.820
FordB	0.781	0.793	<u>0.810</u>	0.798	0.620	0.807	0.711	**0.823**	0.662
Ham	0.657	**0.724**	<u>0.695</u>	0.533	0.467	0.686	0.667	0.667	0.571
LargeKitchenAppliances	0.843	0.789	0.848	0.765	0.795	<u>0.859</u>	0.765	**0.864**	0.811
Phoneme	0.249	0.276	0.289	0.196	0.228	<u>0.321</u>	0.265	**0.382**	0.305
RefrigerationDevices	0.547	0.515	0.517	<u>0.555</u>	0.464	**0.581**	0.499	0.557	0.437
SwedishLeaf	0.925	0.914	<u>0.931</u>	0.925	0.792	0.928	0.922	**0.954**	0.915
UWaveGestureLibraryX	0.806	0.785	0.811	0.784	0.728	0.803	0.762	**0.840**	0.805
Yoga	0.831	0.837	0.878	0.828	0.837	0.818	**0.918**	**0.918**	0.879

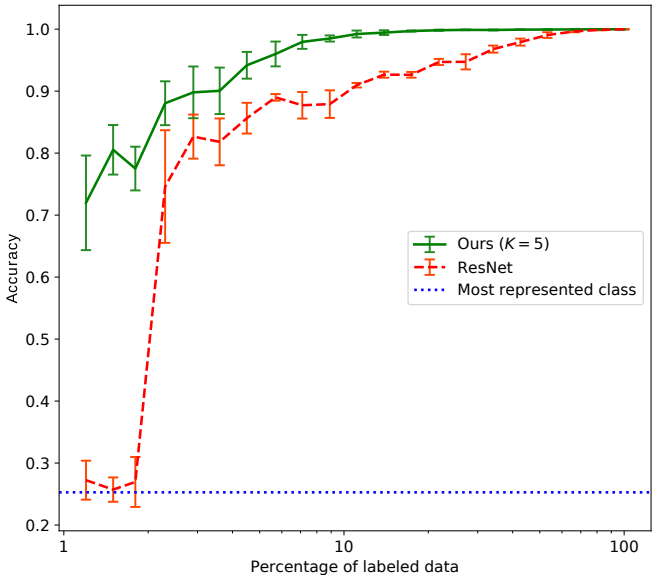
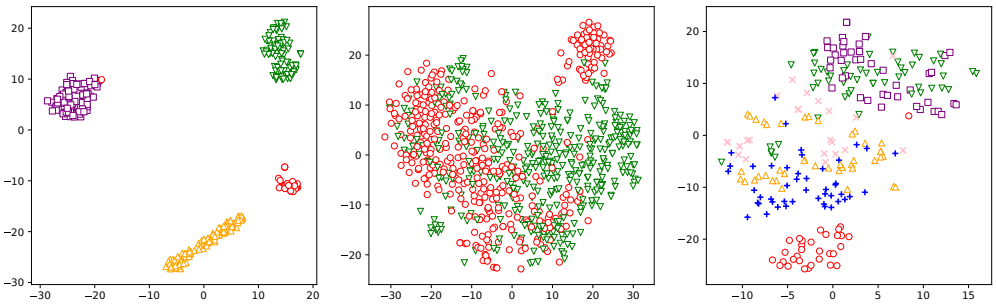


Figure 3.7.: Accuracy of ResNet and our method with respect to the ratio of labeled data on the TwoPatterns dataset. Error bars correspond to the standard deviation over five runs per point for each method.

margin. Moreover, relying on an encoder trained without labels comes with other advantages, as evidenced in the rest of this section.

Sparse labeling. Taking advantage of their unsupervised training, we show that our representations can be successfully used on sparsely labeled datasets compared to supervised methods, since only the SVM is restricted to be learned on the small portion of labeled data. Figure 3.7 shows that an SVM trained on our representations of a randomly chosen labeled set consistently outperforms the supervised neural network ResNet trained on a labeled set of the same size, especially when the percentage of labeled data is small. For example, with only 1.5% of labeled data, we achieve an accuracy of 81%, against only 26% for ResNet, whose performance then equates the one of a random classifier. Moreover, we exceed 99% of accuracy starting from 11% of labeled data, while ResNet only achieves this level of accuracy with more than 50% of labeled data. This shows the relevance of our method in semi-supervised settings, compared to fully supervised methods.

Representations metric space. Besides being suitable for classification purposes, the learned representations may also be used to define a meaningful measure between time series. Indeed, we train, instead of an SVM, a one-nearest-neighbor classifier with respect to the ℓ_2 distance on the same representations, and compare it to DTW, which uses the same classifier on the raw time series. As shown in Appendix A.2 and Table A.1,



(a) DiatomSizeReduction. (b) FordB. (c) OSULeaf.

Figure 3.8.: Two-dimensional t-SNE (van der Maaten and Hinton, 2008) with perplexity 30 of the learned representations of three UCR test sets. Elements classes are distinguishable using their respective marker shapes and colors.

this version of our method outperforms DTW on 66% of the UCR datasets, showing the advantage of the learned representations even in a non-parametric classification setting. We also include quantitative experiments to assess the usefulness of comparing time series using the ℓ_2 distance between their representations with dimensionality reduction (Figure 3.8) and clustering (Figure 3.9) visualizations.

Transferability. We include in the comparisons the classification accuracy for each dataset of an SVM trained on this dataset using the representations computed by an encoder, which was trained on another dataset (FordA, with $K = 5$), to test the transferability of our representations. These scores are reported in Tables 3.1 and A.1. We observe that the scores achieved by this SVM trained on transferred representations are close to the scores reported when the encoder is trained on the same dataset as the SVM, showing the transferability of our representations from a dataset to another, and from time series to other time series with different lengths. More generally, this transferability and the performance of simple classifiers on the representations we learn indicate that they are universal and easy to make use of.

3.5.1.2. Multivariate Time Series

To complement our evaluation on the UCR archive which exclusively contains univariate series, we evaluate our method on multivariate time series. This can be done by simply changing the number of input filters of the first convolutional layer of the proposed encoder. We test our method on all 30 datasets of the UEA archive (Bagnall, Dau, et al., 2018). Full accuracy scores are presented in Appendix A.3, Table A.5.

The UEA archive has been designed as a first attempt to provide a standard archive for multivariate time series classification such as the UCR one for univariate series. As it has only been released recently, we could not compare our method to state-of-the-art classifiers for multivariate time series. However, we provide a comparison

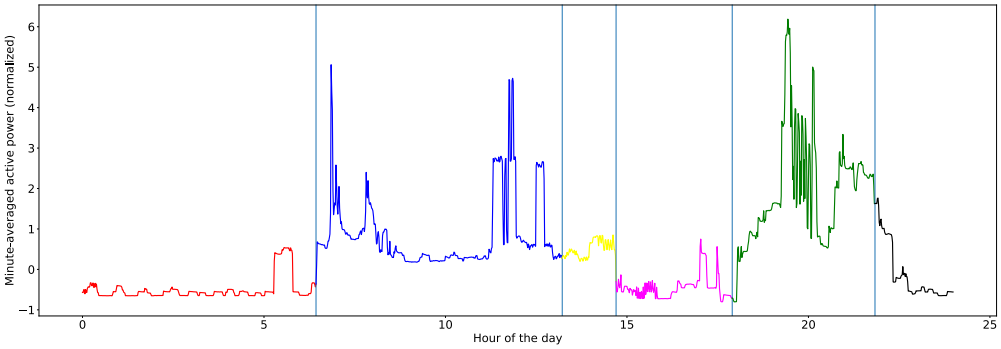


Figure 3.9.: Minute-averaged electricity consumption for a single day from the IHEPC dataset, with respect to the hour of the day. Vertical lines and colors divide the day into six clusters, obtained with k-means clustering based on representations computed on a day-long sliding window. The clustering divides the day into meaningful portions (night, morning, afternoon, evening).

with DTW_D as baseline using results provided by Bagnall, Dau, et al. (2018). DTW_D (dimension-dependent DTW) is an extension of DTW in the multivariate setting and the best baseline studied by Bagnall, Dau, et al. (2018). Overall, our method matches or outperforms DTW_D on 69% of the UEA datasets, which indicates good performance.

3.5.2. Evaluation on Long Time Series

We show the applicability and scalability of our method on long time series without labeling for regression tasks, which could correspond to an industrial application and complements the performed tests on the UCR and UEA archives, whose datasets mostly contain short time series.

The Individual Household Electric Power Consumption (IHEPC) dataset from the UCI Machine Learning Repository (Dheeru and Karra Taniskidou, 2017) is a single time series of length 2 075 259 monitoring the minute-averaged electricity consumption of one French household for four years. We split this time series into training (first 5×10^5 measurements, approximately a year) and testing (remaining measurements) series. The encoder is trained over the training time series on a single Nvidia Tesla P100 GPU in no more than a few hours, showing that our training procedure is scalable to long time series.

We apply the learned encoder on two regression tasks involving two different input scales. We compute, for each time step of the time series, the representations of the last window corresponding to a day (1440 measurements) and a quarter ($12 \cdot 7 \cdot 1440$ measurements) using the same encoder. An example of application of the day-long representations is shown in Figure 3.9. The considered tasks consist in, for each time step, predicting the discrepancy between the mean value of the series for the next period (either a day or quarter) and the one for the previous period. We compare

Table 3.2.: Results obtained on the IHEPC dataset for daily and quarterly forecasting tasks by learning a linear regressor over either our learned representations or the raw time series values.

Task	Metric	Representations	Raw values
Day	Test MSE	$\mathbf{8.92 \times 10^{-2}}$	$\mathbf{8.92 \times 10^{-2}}$
	Wall time	**12 s**	3 min 1 s
Quarter	Test MSE	7.26×10^{-2}	$\mathbf{6.26 \times 10^{-2}}$
	Wall time	**9 s**	1 h 40 min 15 s

linear regressors, trained using gradient descent, to minimize the mean squared error between the prediction and the target, applied either on the raw time series or the previously computed representations.

Results and execution times on an Nvidia Titan Xp GPU are presented in Table 3.2.[4] On both scales of inputs, our representations induce only a slightly degraded performance but provide a large efficiency improvement, due to their small size compared to the raw time series. This shows that a single encoder trained to minimize our time-based loss is able to output representations for different scales of input lengths that are also helpful for other tasks than classification, corroborating their universality.

3.6. Discussion

We briefly discuss in this section several notable aspects of the proposed method.

3.6.1. Behavior of the Learned Representations Throughout Training

Numerical stability. The risk R is defined as the expectation (taken over the random selection of the sequences $\left(\boldsymbol{x}^{\mathrm{ref}}, \boldsymbol{x}^{\mathrm{pos}}, \boldsymbol{x}^{\mathrm{neg}}\right)$) of the loss defined in Equation (3.1). This risk may decrease if all the representations f_θ are scaled by a positive large number. For example, if for some θ_0, for (almost surely) any sequence triplet $\left(\boldsymbol{x}^{\mathrm{ref}}, \boldsymbol{x}^{\mathrm{pos}}, \boldsymbol{x}^{\mathrm{neg}}\right)$, $f_{\theta_0}\left(\boldsymbol{x}^{\mathrm{ref}}\right)^\top f_{\theta_0}(\boldsymbol{x}^{\mathrm{pos}}) \geq 0$ and $f_{\theta_0}\left(\boldsymbol{x}^{\mathrm{ref}}\right)^\top f_{\theta_0}(\boldsymbol{x}^{\mathrm{neg}}) \leq 0$, then:

$$R(\lambda, \theta_0) \triangleq \mathbb{E}_{\boldsymbol{x}^{\mathrm{ref}}, \boldsymbol{x}^{\mathrm{pos}}, \boldsymbol{x}^{\mathrm{neg}}} \left[-\log \sigma\left(\lambda^2 f_{\theta_0}\left(\boldsymbol{x}^{\mathrm{ref}}\right)^\top f_{\theta_0}(\boldsymbol{x}^{\mathrm{pos}})\right) - \log \sigma\left(-\lambda^2 f_{\theta_0}\left(\boldsymbol{x}^{\mathrm{ref}}\right)^\top f_{\theta_0}(\boldsymbol{x}^{\mathrm{neg}})\right)\right] \quad (3.2)$$

[4] While acting on representations of the same size, the quarterly linear regressor is slightly faster than the daily one because the number of quarters in the considered time series is smaller than the number of days.

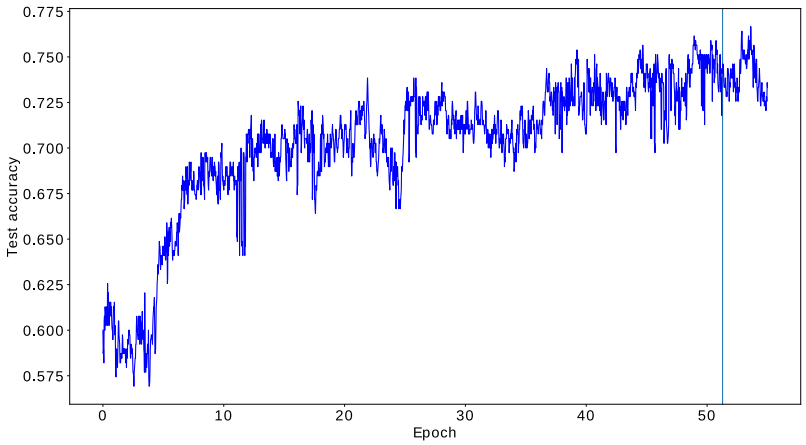


Figure 3.10.: Evolution of the test accuracy during the training of the encoder on the CricketX dataset from the UCR archive (with $K = 10$), with respect to the number of completed epochs. The testing labels were only used for monitoring purposes and the test accuracy was computed after each mini-batch optimization. The vertical line marks the epoch at which 2000 optimization steps were performed, at which point training is stopped in our tests. Test accuracy clearly increases during training.

is a decreasing function of λ, thus λ could diverge to infinity in order to minimize the loss. In other words, the parameters in θ_0 corresponding to the last linear layer could be linearly scaled up, and the representations' norm would indefinitely increase during training. Such a phenomenon is not observed in practice given the capacity of experimented networks, as the mean representation Euclidean norm lies around 20. We posit two possible explanations for this observation: either the condition above is not satisfied (more generally, the loss is not reduced by increasing the representations), or the use of the sigmoid function, that has vanishing gradients, results in an increase of the norm of the representations that is too slow to be observed, or negligible with respect to other weight updates during optimization.

Classification accuracy evolution during training. As shown in Figure 3.10, our unsupervised training clearly makes the classification accuracy of the trained SVM increase with the number of optimization steps.

3.6.2. Influence of K

As mentioned in Section 3.5, the number K of negative examples in the triplet loss can have a significant impact on the performance of the encoder. We notably observe that $K = 1$ leads to slightly lower scores compared to scores obtained when trained with $K > 1$ on the UCR datasets, justifying the use of several negative examples during

Table 3.3.: Results of our training method combined with the proposed causal CNN encoder on the one hand, and with an LSTM encoder on the other hand ($K = 5$). Bold scores indicate the best performing method.

Dataset	Causal CNN	LSTM
Adiac	**0.703**	0.269
Computers	**0.676**	0.492
CricketX	**0.700**	0.136
DistalPhalanxTW	**0.662**	**0.662**
Earthquakes	**0.748**	**0.748**
HandOutlines	**0.908**	0.646
NonInvasiveFetalECGThorax1	**0.904**	0.169
PhalangesOutlinesCorrect	**0.795**	0.613
RefrigerationDevices	**0.547**	0.411
UWaveGestureLibraryX	**0.806**	0.357
Wafer	**0.993**	0.896

training. We did not observe any clear statistical difference between values of K above 1 on the whole archive; however, we notice important differences between different values of K when studying individual datasets. Therefore, we choose to combine several encoders trained with different values of K in order to avoid selecting it as a fixed hyperparameter.

3.6.3. Discussion of the Choice of Encoder

One of the aims of this work is to propose a representation learning method for time series that is scalable. For this reason, and as explained in Section 3.4, we did not consider using an LSTM as encoder f_θ. Nonetheless, we experiment with such an encoder on a small set of UCR datasets in order to get an indication of its performance versus the proposed encoder in this chapter. We use the standard hyperparameters like those used to train the causal CNN encoder and choose a two-layer LSTM with hidden size 256 in order to compare both networks with similar computational time and memory usage. Corresponding results are compiled in Table 3.3.

We observe on this restricted set of experiments that not only does the proposed encoder outperform the LSTM encoder, but it does so by a large margin. This indicates that the proposed causal CNN encoder is more adapted to the considered task and training method.

While we could not produce similar results by replacing the causal CNN with a transformer encoder network, which is also a sequence-to-sequence model but whose introduction was recent at the time we worked on this project, preliminary results indicates that both network options could lead to approximately the same numerical performance. However, using transformer networks significantly remains computationally heavier than using CNNs, thus further motivating our choice of encoder. A

promising future work could consist in investigating the effect of opting for lighter transformer architectures, like sparse transformers (Child et al., 2019).

3.6.4. Reproducibility

Confidence intervals. All UCR datasets are provided with a unique train / test split that we used in our experiments. Compared techniques (DTW, ST, BOSS, HIVE-COTE and EE) are also tested on 100 random train / test splits of these datasets by Bagnall, Lines, et al. (2017) to produce a strong state-of-the-art evaluation, but we do not perform similar resamples as this is beyond the scope of this work and would require significantly more computations. Note that the scores for these methods used in this article are the ones corresponding to the original train / test split of the datasets.

As our method is based on random sampling, the reported scores may vary depending on the random seed. While we do not report standard deviation, the large number of tested datasets prevents large statistical error in the global evaluation of our method. The order of magnitude of accuracy variation between different runs of the combined version of our method is below 0.01. For instance, on four different runs, the corresponding standard variations for, respectively, datasets DiatomSizeReduction, CricketX and UWaveGestureLibraryX are 0.0056, 0.0091 and 0.0053.

Reproducibility report. An extensive reproducibility study (Liljefors, Sorkhei, and Broomé, 2020), conducted subsequently to the publication of this work, shows that the results presented in this chapter are indeed reproducible, thanks to detailed descriptions and provided hyperparameters in Appendix A.1.3. We refer to the corresponding publication for more detail.

3.7. Conclusion

We present an unsupervised representation learning method for time series that is scalable and produces high-quality and easy-to-use embeddings. They are generated by an encoder formed by dilated convolutions that admits variable-length inputs, and trained with an efficient triplet loss using novel time-based negative sampling for time series. Conducted experiments show that these representations are universal and can easily and efficiently be used for diverse tasks such as classification, for which we achieve state-of-the-art performance, and regression.

Part III.

State-Space Predictive Models for Spatiotemporal Data

We tackle in this part the challenging problem of forecasting complex structured high-dimensional data like videos and physical phenomena. We do so through the prism of representation learning, by designing state-space predictive models which, as explained in Section 2.3.2.3, learn complete state representations for each time step of input sequences. This state-space nature, coupled with a temporal model design based on ODEs, allows us to design performant prediction models also offering a meaningful latent space, compared to other autoregressive and recurrent models.

In the next chapter, we focus on predicting the future of videos. This task demanding the prediction model to be stochastic in order to account for the uncertainty of the filmed scene, we propose a novel specifically designed state-space model integrating both stochasticity and links with differential equations to better model natural motion in these videos. This work led to the following publication in an international conference:

> Jean-Yves Franceschi, Edouard Delasalles, Mickaël Chen, Sylvain Lamprier, and Patrick Gallinari (July 2020). "Stochastic Latent Residual Video Prediction". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 3233–3246.

In the second chapter of this part, we more generally consider spatiotemporal data, including videos and physical phenomena, to study spatiotemporal disentanglement, i.e. the separation of spatial and temporal features. While we already consider it in the previous contribution on videos, we explicitly analyze here its signification and suggest grounding it into the formalism of differential equations by interpreting it as a separation of variables in a PDE, yielding a simple and performant disentangled prediction model. This work led to the following publication in an international conference:

> Jérémie Donà, Jean-Yves Franceschi, Sylvain Lamprier, and Patrick Gallinari (2021). "PDE-Driven Spatiotemporal Disentanglement". In: *International Conference on Learning Representations*.

Chapter 4.

Stochastic Latent Residual Video Prediction

4.1. Introduction

Being able to predict the future of a video from a few conditioning frames in a self-supervised manner has many applications in fields such as reinforcement learning (Gregor et al., 2019) or robotics (Babaeizadeh et al., 2018). More generally, it challenges the ability of a model to capture visual and dynamic representations of the world. Video prediction has received a lot of attention from the computer vision community. However, most proposed methods are deterministic, reducing their ability to capture video dynamics, which are intrinsically stochastic (Denton and Fergus, 2018).

Stochastic video prediction is a challenging task that has been tackled by recent works. Most state-of-the-art approaches are based on image-autoregressive models (Denton and Fergus, 2018; Babaeizadeh et al., 2018), built around RNNs like VRNN as explained in Section 2.3.2.3, where each generated frame is fed back to the model to produce the next frame. However, performances of their temporal models innately depend on the capacity of their encoder and decoder, as each generated frame has to be re-encoded in a latent space. Such autoregressive processes induce a high computational cost and strongly tie the frame synthesis and temporal models, which may hurt the performance of the generation process and limit its applicability (Gregor et al., 2019; Rubanova, R. T. Q. Chen, and Duvenaud, 2019).

An alternative approach consists in separating the dynamics of the state representations from the generated frames, which are independently decoded from the latent space: as seen in Section 2.3.2.3), this corresponds to state-space models. In addition to removing the aforementioned link between frame synthesis and temporal dynamics, this is computationally appealing when coupled with a low-dimensional latent space. Moreover, as also mentioned in the introductory chapters, such models can be used to shape a complete representation of the state of a system, e.g. for reinforcement learning applications (Gregor et al., 2019), and are more interpretable than autoregressive models (Rubanova, R. T. Q. Chen, and Duvenaud, 2019). Yet, these state-space models are more difficult to train as they require non-trivial inference schemes (Krishnan, Shalit, and D. Sontag, 2017) and a careful design of the dynamic model (Karl et al., 2017). This leads most successful ones to only be evaluated on small or artificial toy tasks.

In this chapter, we introduce a novel stochastic dynamic model for the task of video prediction which successfully leverages structural and computational advantages of state-space models that operate on low-dimensional latent spaces. Its dynamic component determines the temporal evolution of the system through residual updates of the latent state, conditioned on learned stochastic variables. This formulation allows us to implement an efficient training strategy and process in an interpretable manner complex high-dimensional data such as videos. This residual principle can be linked to the recent advances relating residual networks and ODEs highlighted in Section 2.1.2.2. This interpretation opens new perspectives such as generating videos at different frame rates, as demonstrated in our experiments. The proposed approach outperforms state-of-the-art models on the task of stochastic video prediction, as demonstrated by comparisons with competitive baselines on representative benchmarks.

This chapter is organized as follows. Section 4.2 contextualizes this work beyond Chapter 2 with respect to stochastic video prediction methods as well as state-space and observation-autoregressive predictors, which motivates the introduction of the proposed prediction model specified in Section 4.3. This model is consequently evaluated via thorough experiments in Section 4.4. Finally, the supplementary material referenced in this chapter is given in Appendix B.

4.2. Related Work

Video synthesis covers a range of different tasks, such as video-to-video translation (T.-C. Wang et al., 2018), super-resolution (Caballero et al., 2017), interpolation between distant frames (H. Jiang et al., 2018), generation (Tulyakov et al., 2018), and video prediction, which is the focus of this chapter.

Deterministic models. Inspired by prior sequence generation models using RNNs (Graves, 2013), a number of video prediction methods (Srivastava, Mansimov, and Salakhudinov, 2015; Villegas, J. Yang, Hong, et al., 2017; van Steenkiste et al., 2018; Wichers et al., 2018; Jin et al., 2020) rely on LSTMs, or, like Ranzato et al. (2014), De Brabandere et al. (2016), and J. Xu, Ni, Zefan Li, et al. (2018), on derived networks such as ConvLSTMs (Shi et al., 2015). Indeed, computer vision approaches are usually tailored to high-dimensional video sequences and propose domain-specific techniques such as pixel-level transformations and optical flow (Shi et al., 2015; Walker, Gupta, and Hebert, 2015; Finn, Goodfellow, and Levine, 2016; De Brabandere et al., 2016; Walker, Doersch, et al., 2016; Vondrick and Torralba, 2017; Liang et al., 2017; Z. Liu, Yeh, et al., 2017; Lotter, Kreiman, and Cox, 2017; C. Lu, Hirsch, and Schölkopf, 2017; H. Fan, Linchao Zhu, and Y. Yang, 2019; H. Gao et al., 2019) that help to produce high-quality predictions. Such predictions are, however, deterministic, thus hurting their performance as they fail to generate sharp long-term video frames (Babaeizadeh et al., 2018; Denton and Fergus, 2018). Following M. Mathieu, Couprie, and LeCun (2016), some works proposed to use adversarial losses on the model predictions to sharpen the generated frames (Vondrick and Torralba, 2017; Liang et al., 2017; C. Lu, Hirsch, and Schölkopf, 2017; J. Xu, Ni, and X. Yang, 2018; Yue Wu et al., 2020).

Nonetheless, adversarial losses are notoriously hard to train (Goodfellow, 2016), and lead to mode collapse, thereby preventing diversity of generations.

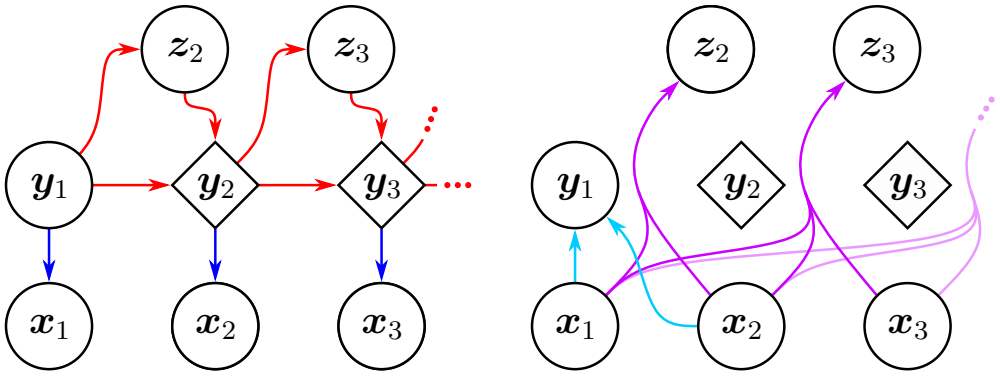
Stochastic and image-autoregressive models. Some approaches rely on exact likelihood maximization, using pixel-level autoregressive generation (van den Oord, Kalchbrenner, et al., 2016; Kalchbrenner et al., 2017; Weissenborn, Täckström, and Uszkoreit, 2020) or normalizing flows through invertible transformations between the observation space and a latent space (Kingma and Dhariwal, 2018; Kumar et al., 2020). However, they require careful design of complex temporal generation schemes manipulating high-dimensional data, thus inducing a prohibitive temporal generation cost.

More efficient continuous models rely on VAEs for the inference of low-dimensional latent state variables. Except Xue et al. (2016) and Z. Xu et al. (2019) who learn a one-frame-ahead VAE, they model sequence stochasticity by incorporating a random latent variable per frame into a deterministic RNN-based image-autoregressive model, like VRNN. Babaeizadeh et al. (2018) integrate stochastic variables into the ConvLSTM architecture of Finn, Goodfellow, and Levine (2016). Concurrently with J. He et al. (2018), Denton and Fergus (2018) use a prior LSTM conditioned on previously generated frames in order to sample random variables that are fed to a predictor LSTM; performances of such methods were improved in follow-up works by increasing networks capacities (Castrejon, Ballas, and Courville, 2019; Villegas, Pathak, et al., 2019). Finally, A. X. Lee, R. Zhang, et al. (2018) combine the ConvLSTM architecture and this learned prior, adding an adversarial loss on the predicted videos to sharpen them at the cost of a diversity drop.

Yet, all these methods are image-autoregressive, as they feed their predictions back into the latent space, thereby tying the frame synthesis and temporal models and increasing their computational cost. Concurrently to our work, Minderer et al. (2019) propose to use the autoregressive VRNN model (Chung, Kastner, et al., 2015) on learned image key-points instead of raw frames. It remains unclear to which extent this change could mitigate the aforementioned problems. We instead tackle these issues by focusing on video dynamics and propose a model that is state-space and acts on a small latent space. This approach yields better experimental results despite weaker video-specific priors.

State-space models. Many latent state-space models have been proposed for sequence modeling (Bayer and Osendorfer, 2014; Fraccaro, S. K. Sønderby, et al., 2016; Fraccaro, Kamronn, et al., 2017; Krishnan, Shalit, and D. Sontag, 2017; Karl et al., 2017; Hafner et al., 2019), usually trained by deep variational inference. These methods, which use locally linear or RNN-based dynamics, are designed for low-dimensional data as learning such models on complex data is challenging, or focus on control or planning tasks.

In contrast, our fully latent method is the first one to be successfully applied to complex high-dimensional data such as videos, thanks to a temporal model based on residual updates of its latent state. It falls within the scope of the recent trend linking differential equations with neural networks in neural network architectures. However,



(a) Generative model p.

(b) Inference model q.

Figure 4.1.: Proposed generative and inference models. Diamonds and circles represent, respectively, deterministic and stochastic states, conditionally to parent states. See also the diagrams of Figures 2.3 and 2.4 for references of other sequential latent variable models.

R. T. Q. Chen, Rubanova, et al. (2018)'s model as well as follow-ups and related works (Rubanova, R. T. Q. Chen, and Duvenaud, 2019; Yıldız, Heinonen, and Lähdesmäki, 2019; Le Guen and Thome, 2020) are either limited to low-dimensional data, prone to overfitting or unable to handle stochasticity within a sequence. Another line of works considers SDEs with neural networks (Ryder et al., 2018; De Brouwer et al., 2019), but are limited to continuous Brownian noise, whereas video prediction additionally requires to model punctual stochastic events.

4.3. Model

We consider the task of stochastic video prediction, consisting in approaching, given a number of conditioning video frames, the distribution of possible future frames.

4.3.1. Latent Residual Dynamic Model

Let $\boldsymbol{x}_{1:T}$ be a sequence of T video frames. We model their evolution by introducing latent variables $\boldsymbol{y}$ that are driven by a dynamic temporal model. Each frame $\boldsymbol{x}_t$ is then generated from the corresponding latent state $\boldsymbol{y}_t$ only, making the dynamics independent of the previously generated frames.

We propose to model the transition function of the latent dynamics of $\boldsymbol{y}$ with a stochastic residual network. State $\boldsymbol{y}_{t+1}$ is chosen to deterministically depend on the previous state $\boldsymbol{y}_t$, conditionally to an auxiliary random variable $\boldsymbol{z}_{t+1}$. These auxiliary variables encapsulate the randomness of the video dynamics. They have a learned factorized Gaussian prior that depends on the previous state only. The model is

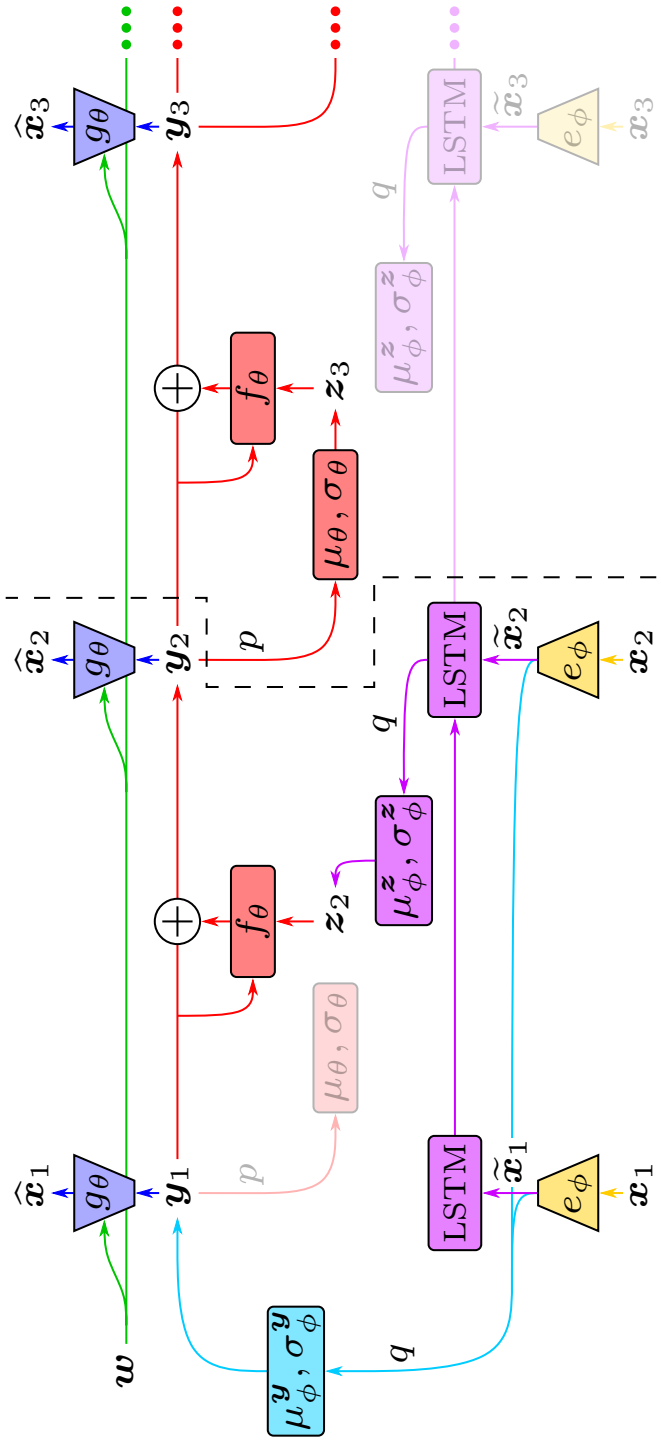


Figure 4.2.: Model architecture with two parts: inference on conditioning frames on the left, generation for extrapolation on the right. The transparent block on the left depicts the prior, and those on the right correspond to the full inference performed at training time. e_ϕ and g_θ are deep CNNs, and other named networks are MLPs.

depicted in Figure 4.1(a), and defined as follows:

$$\begin{cases} \boldsymbol{y}_1 \sim \mathcal{N}(\mathbf{0}, I_d), \\ \boldsymbol{z}_{t+1} \sim \mathcal{N}\big(\mu_\theta(\boldsymbol{y}_t), \sigma_\theta(\boldsymbol{y}_t)\big), \\ \boldsymbol{y}_{t+1} = \boldsymbol{y}_t + f_\theta(\boldsymbol{y}_t, \boldsymbol{z}_{t+1}), \\ \boldsymbol{x}_t \sim \mathcal{G}\big(g_\theta(\boldsymbol{y}_t)\big), \end{cases} \tag{4.1}$$

where μ_θ, σ_θ, f_θ and g_θ are θ-parameterized neural networks, and $\mathcal{G}\big(g_\theta(\boldsymbol{y}_t)\big)$ is a probability distribution parameterized by $g_\theta(\boldsymbol{y}_t)$. In our experiments, $\mathcal{G}$ is a normal distribution with mean $g_\theta(\boldsymbol{y}_t)$ and constant diagonal variance. Note that $\boldsymbol{y}_1$ is assumed to have a standard Gaussian prior, and, in our VAE setting, will be inferred from conditioning frames for the prediction task, as shown in Section 4.3.3.

The residual update rule takes inspiration in the Euler discretization scheme of differential equations. The state of the system $\boldsymbol{y}_t$ is updated by its first-order movement, i.e. the residual $f_\theta(\boldsymbol{y}_t, \boldsymbol{z}_{t+1})$. Compared to a regular RNN, this simple principle makes our temporal model lighter and more interpretable. Equation (4.1), however, differs from a discretized ODE because of the introduction of the stochastic discrete-time variables $\boldsymbol{z}$. Nonetheless, we propose to allow the Euler step size Δt to be smaller than 1, as a way to make the temporal model closer to a continuous dynamics. The updated dynamics becomes, with $1/\Delta t \in \mathbb{N}$ to synchronize the step size with the video frame rate:

$$\boldsymbol{y}_{t+\Delta t} = \boldsymbol{y}_t + \Delta t \cdot f_\theta\Big(\boldsymbol{y}_t, \boldsymbol{z}_{\lfloor t \rfloor + 1}\Big). \tag{4.2}$$

For this formulation, the auxiliary variable $\boldsymbol{z}_t$ is kept constant between two integer time steps. Note that a different Δt can be used during training or testing. This allows our model to generate videos at an arbitrary frame rate since each intermediate latent state can be decoded in the observation space. This ability enables us to observe the quality of the learned dynamic as well as challenge its ODE inspiration by testing its generalization to the continuous limit in Section 4.4. In the following, we consider Δt as a hyperparameter. For the sake of clarity, we consider that $\Delta t = 1$ in the rest of this section; generalizing to a smaller Δt is straightforward as Figure 4.1(a) remains unchanged.

Let us finally highlight the decomposition of the dynamics in Equation (4.1), involving, besides the stochastic variables $\boldsymbol{z}$, the conditionally deterministic $\boldsymbol{y}$. This is unlike the deep Markov model of Figure 2.3(a), where model states are stochastic. A simpler alternative, closer to the latter, would be to choose:

$$(\boldsymbol{y}_{t+1} - \boldsymbol{y}_t) \sim \mathcal{N}\big(\mu_\theta(\boldsymbol{y}_t), \sigma_\theta(\boldsymbol{y}_t)\big), \tag{4.3}$$

which resembles the Euler-Maruyama approximation scheme of SDEs (Kloeden and Platen, 1992). Besides this connection to SDEs which are unfit for video modeling as argued in Section 4.2, Karl et al. (2017) suggest that such direct stochastic transition penalizes the prediction ability of the system, which we experimentally confirmed by preliminary experiments. In contrast, our choice of Equation (4.1) circumvents this issue because it constrains the inference model, presented in Section 4.3.3, to backpropagate a gradient through the whole sequence of states $\boldsymbol{y}$ by the deterministic

transitions, in opposition to the deep Markov model of Figure 2.4(a). This can actually be retrieved in the VRNN model of Figure 2.3(b) as well, although in a different setting and without the state-space assumption, where the RNN hidden states $\boldsymbol{h}$ ensure the presence of such a gradient path.

4.3.2. Content Variable

Some components of video sequences can be static, such as the background or shapes of moving objects, and may not impact the dynamics. Therefore, we model them separately, similarly to Denton and Birodkar (2017), Yingzhen and Mandt (2018) and other works leveraging spatiotemporal disentanglement as seen in Section 2.2.2.2. We compute a content variable $\boldsymbol{w}$ that remains constant throughout the whole generation process and is fed together with $\boldsymbol{y}_t$ into the frame generator. It enables the dynamical part of the model to focus only on movement, hence being lighter and more stable. Moreover, it allows us to leverage architectural advances in neural networks like skip connections (Ronneberger, Fischer, and Brox, 2015) to produce more realistic frames.

This content variable is a deterministic ψ-parameterized function c_ψ of a fixed number $k < T$ of frames $\boldsymbol{x}_{\mathrm{c}}^{(k)} = \left(\boldsymbol{x}_{i_1}, \dots, \boldsymbol{x}_{i_k}\right)$:

$$\begin{cases} \boldsymbol{w} = c_\psi\left(\boldsymbol{x}_{\mathrm{c}}^{(k)}\right) = c_\psi\left(\boldsymbol{x}_{i_1}, \dots, \boldsymbol{x}_{i_k}\right) \\ \boldsymbol{x}_t \sim \mathcal{G}\left(g_\theta(\boldsymbol{y}_t, \boldsymbol{w})\right). \end{cases} \tag{4.4}$$

During testing, $\boldsymbol{x}_{\mathrm{c}}^{(k)}$ are the last k conditioning frames (usually between 2 and 5).

This content variable is not endowed with any probabilistic prior, contrary to the dynamic variables $\boldsymbol{y}$ and $\boldsymbol{z}$. Thus, the information it contains is not constrained in the loss function (see Section 4.3.3), but only architecturally. To prevent temporal information from leaking in $\boldsymbol{w}$, we design c_ψ as a permutation-invariant function (Zaheer et al., 2017), consisting in an Multilayer Perceptron (MLP) fed with the sum of individual frame representations, following Santoro et al. (2017). Additionally, we propose to uniformly sample the k encoded frames within $\boldsymbol{x}_{1:T}$ during training.

This absence of prior and its architectural constraint allow $\boldsymbol{w}$ to contain as much non-temporal information as possible while preventing it from containing dynamic information. On the other hand, due to their strong standard Gaussian priors, $\boldsymbol{y}$ and $\boldsymbol{z}$ are encouraged to discard unnecessary static information. Therefore, $\boldsymbol{y}$ and $\boldsymbol{z}$ should only contain temporal information that could not be captured by $\boldsymbol{w}$. Note that this content variable can be removed from our model, yielding a more classical deep state-space model; an experiment in this setting is presented in Appendix B.5.

4.3.3. Variational Inference and Architecture

Following the generative process depicted in Figure 4.1(a), the conditional joint probability of the full model, given a content variable $\boldsymbol{w}$, can be written as:

$$p(\boldsymbol{x}_{1:T}, \boldsymbol{z}_{2:T}, \boldsymbol{y}_{1:T} \mid \boldsymbol{w}) = p(\boldsymbol{y}_1) \prod_{t=2}^{T} p(\boldsymbol{z}_t, \boldsymbol{y}_t \mid \boldsymbol{y}_{t-1}) \prod_{t=1}^{T} p(\boldsymbol{x}_t \mid \boldsymbol{y}_t, \boldsymbol{w}), \tag{4.5}$$

with:

$$p(\boldsymbol{z}_t, \boldsymbol{y}_t \mid \boldsymbol{y}_{t-1}) = p(\boldsymbol{z}_t \mid \boldsymbol{y}_{t-1}) p(\boldsymbol{y}_t \mid \boldsymbol{y}_{t-1}, \boldsymbol{z}_t). \tag{4.6}$$

According to the expression of $\boldsymbol{y}_{t+1}$ in Equation (4.1):

$$p(\boldsymbol{y}_t \mid \boldsymbol{y}_{t-1}, \boldsymbol{z}_t) = \delta\big(\boldsymbol{y}_t - \boldsymbol{y}_{t-1} - f_\theta(\boldsymbol{y}_{t-1}, \boldsymbol{z}_t)\big), \tag{4.7}$$

where δ is the Dirac delta function centered on $\mathbf{0}$. Hence, in order to optimize the likelihood of the observed videos $p(\boldsymbol{x}_{1:T} \mid \boldsymbol{w})$, we need to infer latent variables $\boldsymbol{y}_1$ and $\boldsymbol{z}_{2:T}$. This is done by deep variational inference using the inference model parameterized by ϕ and shown in Figure 4.1(b), which comes down to considering a variational distribution $q_{Z,Y}$ defined and factorized as follows:

$$q_{Z,Y} \triangleq q(\boldsymbol{z}_{2:T}, \boldsymbol{y}_{1:T} \mid \boldsymbol{x}_{1:T}, \boldsymbol{w}) = q(\boldsymbol{y}_1 \mid \boldsymbol{x}_{1:k}) \prod_{t=2}^{T} q(\boldsymbol{z}_t \mid \boldsymbol{x}_{1:t}) \underbrace{q(\boldsymbol{y}_t \mid \boldsymbol{y}_{t-1}, \boldsymbol{z}_t)}_{=p\left(\boldsymbol{y}_t \mid \boldsymbol{y}_{t-1}, \boldsymbol{z}_t\right)}, \tag{4.8}$$

with $q(\boldsymbol{y}_t \mid \boldsymbol{y}_{t-1}, \boldsymbol{z}_t) = p(\boldsymbol{y}_t \mid \boldsymbol{y}_{t-1}, \boldsymbol{z}_t)$ being the aforementioned Dirac delta function. This yields the following ELBO, whose full derivation starting from the general ELBO of Equation (2.22) is provided in Appendix B.1:

$$\begin{aligned} &\log p(\boldsymbol{x}_{1:T} \mid \boldsymbol{w}) \geq \mathcal{L}(\boldsymbol{x}_{1:T}; \boldsymbol{w}, \theta, \phi) \\ &\triangleq \mathbb{E}_{(\boldsymbol{z}_{2:T}, \boldsymbol{y}_{1:T}) \sim q_{Z,Y}} \left[\sum_{t=1}^{T} \log p(\boldsymbol{x}_t \mid \boldsymbol{y}_t, \boldsymbol{w}) - \sum_{t=2}^{T} D_{\mathrm{KL}}\big(q(\boldsymbol{z}_t \mid \boldsymbol{x}_{1:t}) \,\big\|\, p(\boldsymbol{z}_t \mid \boldsymbol{y}_{t-1})\big) \right] \\ &\qquad\qquad - D_{\mathrm{KL}}\big(q(\boldsymbol{y}_1 \mid \boldsymbol{x}_{1:k}) \,\big\|\, p(\boldsymbol{y}_1)\big). \end{aligned} \tag{4.9}$$

Note that the dependency of p and q in parameters θ and ϕ is hidden in the previous equations, for the sake of clarity; in particular, while p only depends on θ, q depends on both θ and ϕ since the transition function in Equation (4.8) between $\boldsymbol{y}_t$ and $\boldsymbol{z}_{t+1}$ on the one hand and $\boldsymbol{y}_{t+1}$ on the other hand is shared by p and q.

The sum of KLD expectations implies considering the full past sequence of inferred states for each time step, due to the dependence on conditionally deterministic variables $\boldsymbol{y}_{2:T}$. However, optimizing $\mathcal{L}(\boldsymbol{x}_{1:T}; \boldsymbol{w}, \theta, \phi)$ with respect to model parameters θ and variational parameters ϕ can be done efficiently by sampling a single full sequence of states from $q_{Z,Y}$ per example, and computing gradients by backpropagation trough all inferred variables, using the reparameterization trick (Kingma and Welling, 2014). We classically choose $q(\boldsymbol{y}_1 \mid \boldsymbol{x}_{1:k})$ and $q(\boldsymbol{z}_t \mid \boldsymbol{x}_{1:t})$ to be factorized Gaussian so that all KLDs can be computed analytically.

We include an ℓ_2 regularization term on residuals f_θ applied to $\boldsymbol{y}$ which stabilizes the temporal dynamics of the residual network, as noted by Behrmann et al. (2019), Rousseau, Drumetz, and Fablet (2019), and de Bézenac, Ayed, and Gallinari (2021). Given a set of videos $\mathcal{X}$, the full optimization problem, where $\mathcal{L}$ is defined as in

Equation (4.9), is then given as:

$$\arg\max_{\theta,\phi,\psi} \sum_{\boldsymbol{x}\in\mathcal{X}} \Bigg[\mathbb{E}_{\boldsymbol{x}_{\mathrm{c}}^{(k)}} \mathcal{L}\Big(\boldsymbol{x}_{1:T}; c_\psi\Big(\boldsymbol{x}_{\mathrm{c}}^{(k)}\Big), \theta, \phi\Big) - \lambda \cdot \mathbb{E}_{(\boldsymbol{z}_{2:T}, \boldsymbol{y}_{1:T})\sim q_{Z,Y}} \sum_{t=2}^{T} \big\| f_\theta(\boldsymbol{y}_{t-1}, \boldsymbol{z}_t) \big\|_2 \Bigg]. \tag{4.10}$$

Figure 4.2 depicts the full architecture of our temporal model, also showing how the model is applied during testing. The first latent variables are inferred with the conditioning frames and are then predicted with the dynamic model. In contrast, during training, each frame of the input sequence is considered for inference, which is done as follows. Firstly, each frame $\boldsymbol{x}_t$ is independently encoded into a vector-valued representation $\widetilde{\boldsymbol{x}}_t$, with $\widetilde{\boldsymbol{x}}_t = e_\phi(\boldsymbol{x}_t)$. $\boldsymbol{y}_1$ is then inferred using an MLP on the first k encoded frames $\widetilde{\boldsymbol{x}}_{1:k}$. Each $\boldsymbol{z}_t$ is inferred in a feed-forward fashion with an LSTM on the encoded frames. Inferring $\boldsymbol{z}$ with such a filtering procedure experimentally performs better than smoothing by inferring them from the whole sequence $\boldsymbol{x}_{1:T}$; we hypothesize that this follows from the fact that this filtering scheme is closer to the prediction setting, where the future is not available.

4.4. Experiments

This section exposes the experimental results of our method on four standard stochastic video prediction datasets.[1] We compare our method with state-of-the-art baselines on stochastic video prediction. Furthermore, we qualitatively study the dynamics and latent space learned by our model.[2]

We used Python 3.7.6 and PyTorch 1.4.0 (Paszke et al., 2019) to implement our model. Each model was trained on Nvidia GPUs with CUDA 10.1 using mixed-precision training (Micikevicius et al., 2018) thanks to the Apex library.[3] Training details are described in Appendix B.3.

4.4.1. Evaluation and Comparisons

The stochastic nature and novelty of the task of stochastic video prediction make it challenging to evaluate (A. X. Lee, R. Zhang, et al., 2018): since videos and models are stochastic, comparing the ground truth and a predicted video is not adequate. We thus adopt the common approach (Denton and Fergus, 2018; A. X. Lee, R. Zhang, et al., 2018) consisting in, for each test sequence, sampling from the tested model a given number (here, 100) of possible futures and reporting the best performing sample with respect to the true video. We report this measure for three commonly used

[1] Code and datasets are available at `https://github.com/edouardelasalles/srvp`. Pretrained models are downloadable at `https://data.lip6.fr/srvp/`.
[2] Animated video samples are available at `https://sites.google.com/view/srvp/`.
[3] `https://github.com/nvidia/apex`.

metrics that are computed frame-wise and averaged over time: Peak Signal-to-Noise Ratio (PSNR) (higher is better), Structured Similarity (SSIM) (higher is better), and Learned Perceptual Image Patch Similarity (LPIPS) (lower is better, R. Zhang et al., 2018). PSNR greatly penalizes errors in predicted dynamics, as it is a pixel-level measure derived from the ℓ_2 distance, but might also favor blurry predictions. SSIM rather compares local frame patches to circumvent this issue but loses some dynamics information. LPIPS compares images through a learned distance between activations of deep CNNs trained on image classification tasks and has been shown to better correlate with human judgment on real images. Finally, the recently proposed Fréchet Video Distance (FVD) (lower is better, Unterthiner et al., 2018) aims at directly comparing the distribution of predicted videos with the ground truth distribution through the representations computed by a deep CNN trained on action recognition tasks. It has been shown, independently of LPIPS, to better capture the realism of predicted videos than PSNR and SSIM. We treat all four metrics as complementary, as they capture different scales and modalities.

We present experimental results on a simulated dataset and three real-world datasets, that we briefly present in the following and detail in Appendix B.2. For the sake of concision, we only display a handful of qualitative samples in this section and refer to Appendix B.6 and our website for additional samples. We compare our model against several variational state-of-the-art models: SV2P (Babaeizadeh et al., 2018), SVG (Denton and Fergus, 2018), SAVP (A. X. Lee, R. Zhang, et al., 2018), and StructVRNN (Minderer et al., 2019). Note that SVG has the closest training and architecture to ours among the state of the art. Therefore, we use the same neural architecture as SVG for our encoders and decoders in order to perform fair comparisons with this method.

All baseline results are presented only on the datasets on which they were tested in the original articles. They are obtained with pretrained models released by the authors, except those of SVG on the Moving MNIST dataset and StructVRNN on the Human3.6M dataset, for which we train models using the code and hyperparameters provided by the authors (see Appendix B.2). Note that we choose the learned prior version of SVG for all datasets but KTH, for which we select the fixed prior version, as done by its authors (Denton and Fergus, 2018). Unless specified otherwise, our model is tested with the same Δt as in training (see Equation (4.2)).

4.4.2. Datasets and Prediction Results

4.4.2.1. Stochastic Moving MNIST

This dataset consists of one or two MNIST digits (LeCun, Bottou, et al., 1998) moving linearly and randomly bouncing on walls with new direction and velocity sampled randomly at each bounce (Denton and Fergus, 2018).

Figure 4.4 (left) shows quantitative results with two digits. Our model outperforms SVG on both PSNR and SSIM; LPIPS and FVD are not reported as they are not relevant for this synthetic task. Decoupling dynamics from image synthesis allows our method to maintain temporal consistency despite high-uncertainty frames where crossing digits become indistinguishable. For instance in Figure 4.3, the digits shapes

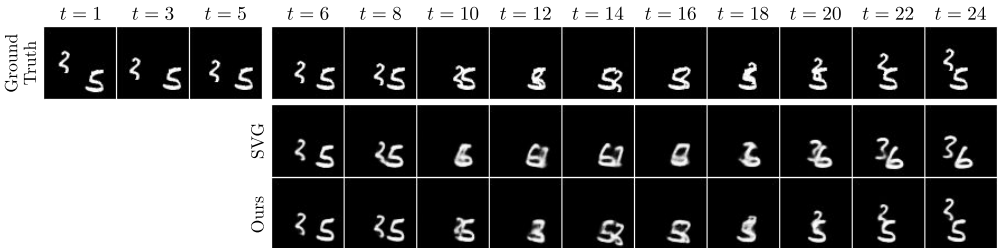


Figure 4.3.: Conditioning frames and corresponding ground truth and best samples with respect to PSNR from SVG and our method for an example of the Stochastic Moving MNIST dataset.

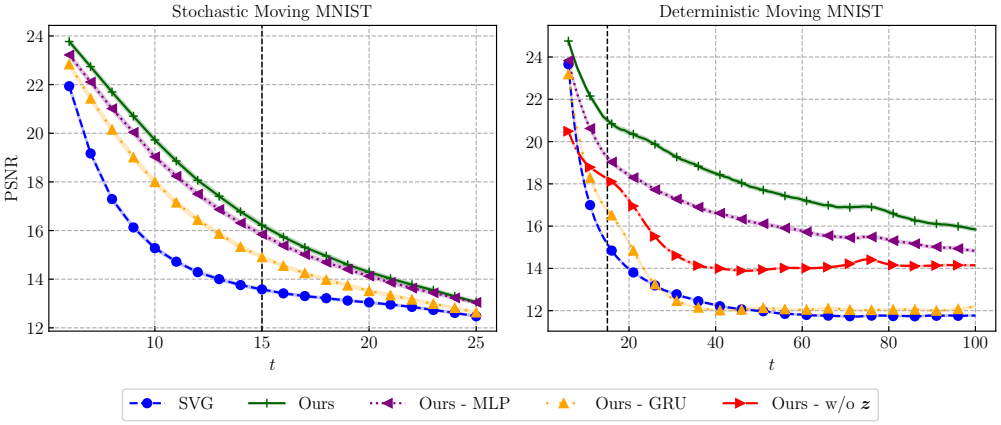


Figure 4.4.: Mean PSNR scores with respect to t for all tested models on the Moving MNIST dataset, with their 95%-confidence intervals. Dark vertical bars mark the length of training sequences.

Table 4.1.: Time-averaged numerical results (mean and 95%-confidence interval) for PSNR and SSIM of tested methods on the two-digits Moving MNIST dataset. Bold scores indicate the best performing method for each metric and, where appropriate, scores whose means lie in the confidence interval of the best performing method.

Models	Stochastic		Deterministic	
	PSNR	SSIM	PSNR	SSIM
SVG	14.50 ± 0.04	0.7090 ± 0.0015	12.85 ± 0.03	0.6185 ± 0.0011
Ours	$\mathbf{16.93 \pm 0.07}$	$\mathbf{0.7799 \pm 0.0020}$	$\mathbf{18.25 \pm 0.06}$	$\mathbf{0.8300 \pm 0.0017}$
Ours - MLP	16.55 ± 0.06	0.7694 ± 0.0019	16.70 ± 0.05	0.7876 ± 0.0015
Ours - GRU	15.80 ± 0.05	0.7464 ± 0.0016	13.17 ± 0.03	0.6237 ± 0.0011
Ours - w/o $\boldsymbol{z}$	—	—	14.99 ± 0.03	0.4757 ± 0.0019

Table 4.2.: FVD scores for all tested methods on the KTH, Human3.6M and BAIR datasets with their 95%-confidence intervals over five different samples from the models. Bold scores indicate the best performing method for each dataset.

Method	KTH	Human3.6M	BAIR
SV2P	636 ± 1	—	965 ± 17
SAVP	374 ± 3	—	$\mathbf{152 \pm 9}$
SVG	377 ± 6	—	255 ± 4
StructVRNN	—	556 ± 9	—
Ours	$\mathbf{222 \pm 3}$	$\mathbf{416 \pm 5}$	163 ± 4
Ours - $\frac{\Delta t}{2}$	244 ± 3	$\mathbf{415 \pm 3}$	222 ± 42
Ours - MLP	255 ± 4	582 ± 4	162 ± 4
Ours - GRU	240 ± 5	1050 ± 20	178 ± 10

change after they cross in the SVG prediction, while our model predicts the correct digits. To evaluate the predictive ability on a longer horizon, we perform experiments on the deterministic version of the dataset (Srivastava, Mansimov, and Salakhudinov, 2015) with only one prediction per model to compute PSNR and SSIM. We show the results up to $t + 95$ in Figure 4.4 (right). We can see that our model better captures the dynamics of the problem compared to SVG as its performance decreases significantly less over time, especially at a long-term horizon.

We also compare to two alternative versions of our model in Figure 4.4, where the residual dynamic function is replaced with an MLP or a GRU. Our residual model outperforms these two versions on both modalities of the dataset, especially on the deterministic version of the latter, showing its intrinsic advantage at modeling long-term dynamics. Finally, on the deterministic version of Moving MNIST, we compare to an alternative version of our method where $\boldsymbol{z}$ is entirely removed, resulting in a temporal model close to the one presented by R. T. Q. Chen, Rubanova, et al. (2018). The loss of performance of this alternative model is significant, showing that our stochastic residual sequential variable model offers a substantial advantage even when used in a deterministic environment.

4.4.2.2. KTH Action Dataset

This dataset is composed of real-world videos of people performing a single action per video in front of different backgrounds (Schüldt, Laptev, and Caputo, 2004). Uncertainty lies in the appearance of subjects, the actions they perform, and how they are performed.

We substantially outperform on this dataset every considered baseline for each metric, as shown in Figure 4.5 and Table 4.2. In some videos, the subject only appears after the conditioning frames, requiring the model to sample the moment and location of the subject appearance, as well as its action. This critical case is illustrated in Figure 4.6.

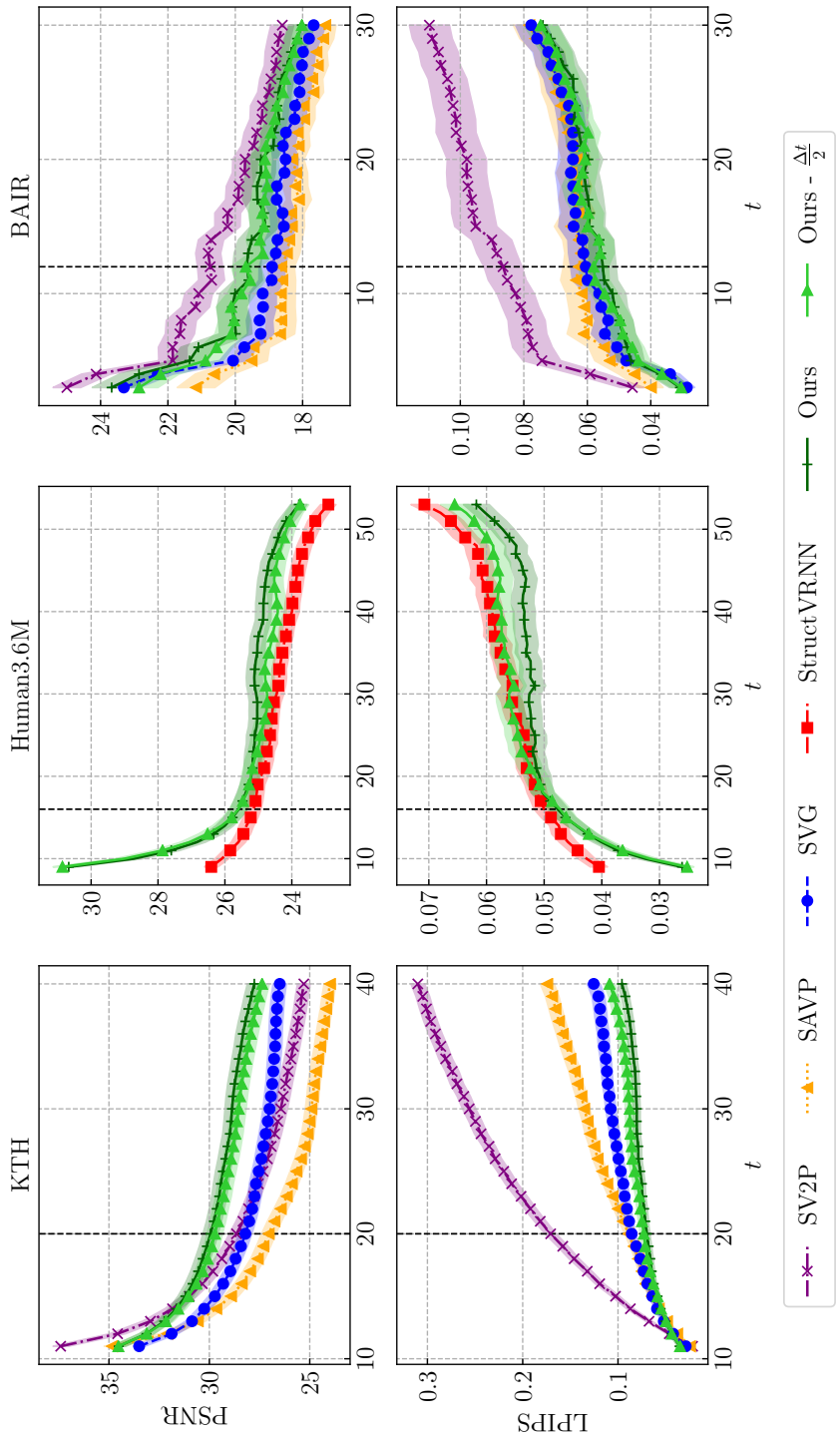


Figure 4.5.: PSNR and LPIPS scores with respect to t for all tested models on the KTH (left column), Human3.6M (center) and BAIR (right) datasets, with their 95%-confidence intervals. Dark vertical bars mark the length of training sequences.

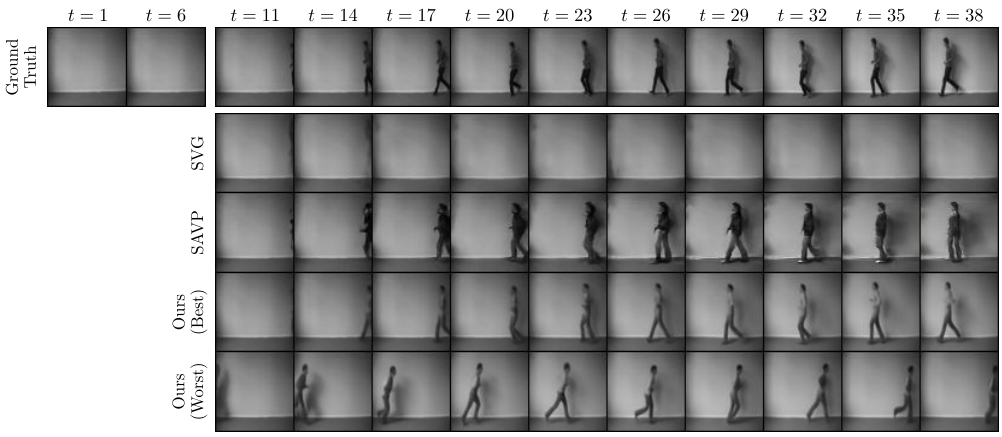


Figure 4.6.: Conditioning frames and corresponding ground truth, best samples from SVG, SAVP and our method, and worst sample from our method, for a video of the KTH dataset. Samples are chosen according to their LPIPS with respect to the ground truth. SVG fails to make a person appear, unlike SAVP and our model. The latter better predicts the subject pose and produces more realistic predictions.

Table 4.3.: Time-averaged numerical results (mean and 95%-confidence interval, when relevant) for PSNR, SSIM, and LPIPS of tested methods on the KTH dataset. Bold scores indicate the best performing method for each metric and, where appropriate, scores whose means lie in the confidence interval of the best performing method.

Models	PSNR	SSIM	LPIPS
SV2P	28.19 ± 0.31	0.8141 ± 0.0050	0.2049 ± 0.0053
SAVP	26.51 ± 0.29	0.7564 ± 0.0062	0.1120 ± 0.0039
SVG	28.06 ± 0.29	0.8438 ± 0.0054	0.0923 ± 0.0038
Ours	$\mathbf{29.69 \pm 0.32}$	$\mathbf{0.8697 \pm 0.0046}$	$\mathbf{0.0736 \pm 0.0029}$
Ours - $\frac{\Delta t}{2}$	$\mathbf{29.43 \pm 0.33}$	0.8633 ± 0.0049	0.0790 ± 0.0034
Ours - MLP	28.91 ± 0.34	0.8527 ± 0.0051	0.0799 ± 0.0032
Ours - GRU	29.14 ± 0.33	0.8590 ± 0.0050	0.0790 ± 0.0032

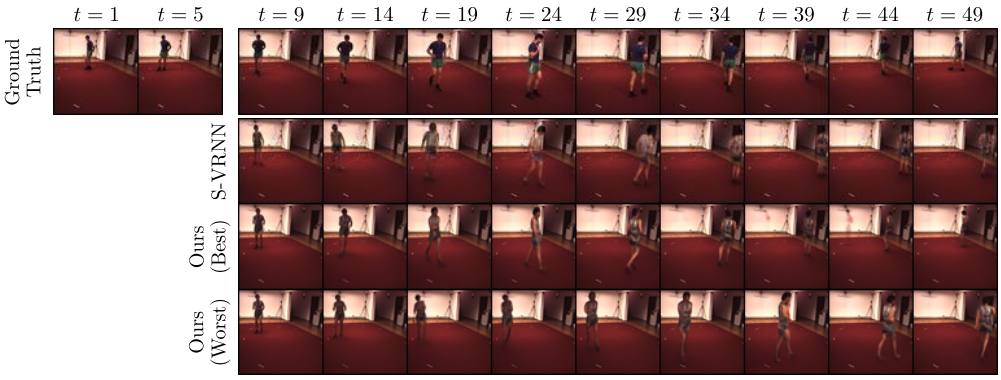


Figure 4.7.: Conditioning frames and corresponding ground truth, best samples from StructVRNN and our method, and worst sample from our method, with respect to LPIPS, for a video of the Human3.6M dataset. Our method better captures the dynamics of the subject and produces less artefacts than StructVRNN.

There, SVG fails to even generate a moving person; only SAVP and our model manage to do so, and our best sample is closer to the subject's poses compared to SAVP. Moreover, the worst sample of our model demonstrates that it captures the diversity of the dataset by making a person appear at different time steps and with different speeds. An additional experiment on this dataset in Section 4.4.3 studies the influence of the encoder and decoder architecture on SVG and our model.

Finally, Tables 4.2 and 4.3 compare our method to its MLP and GRU alternative versions, leading to two conclusions. Firstly, it confirms the structural advantage of residual dynamics observed on Moving MNIST. Indeed, both MLP and GRU lose on all metrics, and especially in terms of realism according to LPIPS and FVD. Secondly, all three versions of our model (residual, MLP, GRU) outperform prior methods. Therefore, this improvement is due to their common inference method, latent nature and content variable, strengthening our motivation to propose a non-autoregressive model.

4.4.2.3. Human3.6M

This dataset is also made of videos of subjects performing various actions (Ionescu, F. Li, and Sminchisescu, 2011; Ionescu, Papava, et al., 2014). While there are more actions and details to capture with fewer training subjects than in KTH, the video backgrounds are less varied, and subjects always remain within the frames.

As reported in Figure 4.5 and Table 4.2, we significantly outperform, with respect to all considered metrics, StructVRNN, which is the state of the art on this dataset and has been shown to surpass both SAVP and SVG by Minderer et al. (2019). Figure 4.7 shows the challenges of the dataset; in particular, both methods do not correctly capture the subject appearance. Nonetheless, our model better captures its movements

Table 4.4.: Time-averaged numerical results (mean and 95%-confidence interval, when relevant) for PSNR, SSIM, and LPIPS of tested methods on the Human3.6M dataset. Bold scores indicate the best performing method for each metric and, where appropriate, scores whose means lie in the confidence interval of the best performing method.

Models	PSNR	SSIM	LPIPS
StructVRNN	24.46 ± 0.17	0.8868 ± 0.0025	0.0557 ± 0.0013
Ours	$\mathbf{25.30 \pm 0.19}$	$\mathbf{0.9074 \pm 0.0022}$	$\mathbf{0.0509 \pm 0.0013}$
Ours - $\frac{\Delta t}{2}$	$\mathbf{25.14 \pm 0.21}$	0.9049 ± 0.0024	0.0534 ± 0.0015
Ours - MLP	25.00 ± 0.19	0.9047 ± 0.0021	0.0529 ± 0.0013
Ours - GRU	23.54 ± 0.18	0.8868 ± 0.0022	0.0683 ± 0.0014

and produces more realistic frames.

Comparisons to the MLP and GRU versions demonstrate once again the advantage of using residual dynamics. GRU obtains low scores on all metrics, which is coherent with similar results for SVG reported by Minderer et al. (2019). While the MLP version remains close to the residual model on PSNR, SSIM and LPIPS, it is largely beaten by the latter in terms of FVD.

4.4.2.4. BAIR Robot Pushing Dataset

This dataset contains videos of a Sawyer robotic arm pushing objects on a tabletop (Ebert et al., 2017). It is highly stochastic as the arm can change its direction at any moment. Our method achieves similar or better results compared to state-of-the-art models in terms of PSNR, SSIM and LPIPS, as shown in Figure 4.5, except for SV2P that produces very blurry samples, as seen in Appendix B.6, yielding good PSNR but prohibitive LPIPS scores. Our method obtains the second-best FVD score, close to SAVP whose adversarial loss enables it to better model small objects, and outperforms SVG, whose variational architecture is closest to ours, demonstrating the advantage of non-autoregressive methods. Recent advances (Villegas, Pathak, et al., 2019) indicate that performance of such variational models can be improved by increasing networks capacities, but this is out of the scope of this work.

4.4.3. Illustration of Residual, State-Space and Latent Properties

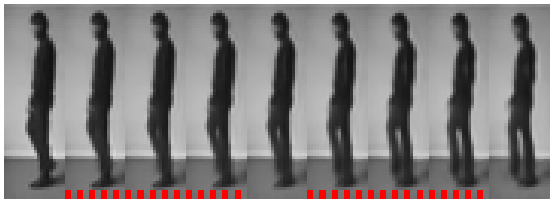
We qualitatively and quantitatively illustrate in the rest of this section the advantages of the peculiar properties of our video prediction model introduced in the previous sections.

4.4.3.1. Generation at Varying Frame Rate

We challenge here the ODE inspiration of our model. Equation (4.2) amounts to learning a residual function $f_{\boldsymbol{z}_{\lfloor t \rfloor + 1}}$ over $t \in \big[\lfloor t \rfloor, \lfloor t \rfloor + 1\big)$. We aim at testing whether

Table 4.5.: Time-averaged numerical results (mean and 95%-confidence interval, when relevant) with respect to PSNR, SSIM, and LPIPS of tested methods on the BAIR dataset. Bold scores indicate the best performing method for each metric and, where appropriate, scores whose means lie in the confidence interval of the best performing method.

Models	PSNR	SSIM	LPIPS
SV2P	**20.39 ± 0.27**	**0.8169 ± 0.0086**	0.0912 ± 0.0053
SAVP	18.44 ± 0.25	0.7887 ± 0.0092	0.0634 ± 0.0026
SVG	18.95 ± 0.26	0.8058 ± 0.0088	0.0609 ± 0.0034
Ours	19.59 ± 0.27	**0.8196 ± 0.0084**	**0.0574 ± 0.0032**
Ours - $\frac{\Delta t}{2}$	19.45 ± 0.26	**0.8196 ± 0.0082**	**0.0579 ± 0.0032**
Ours - MLP	19.56 ± 0.26	**0.8194 ± 0.0084**	**0.0572 ± 0.0032**
Ours - GRU	19.41 ± 0.26	**0.8170 ± 0.0084**	**0.0585 ± 0.0032**



(a) Cropped KTH sample (training $\Delta t = 1/2$).

(b) Cropped Human3.6M sample (training $\Delta t = 1/2$).

(c) Cropped BAIR sample (training $\Delta t = 1$).

Figure 4.8.: Generation examples at doubled or quadrupled frame rate, using a halved Δt compared to training. Frames including a bottom red dashed bar are intermediate frames.

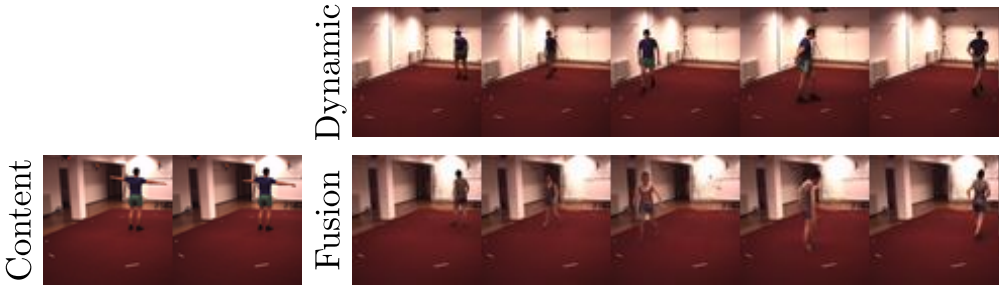


Figure 4.9.: Video (bottom right) generated from the dynamic latent state $\boldsymbol{y}$ inferred with a video (top) and the content variable $\boldsymbol{w}$ computed with the conditioning frames of another video (left). The generated video keeps the same background as the bottom left frames, while the subject moves accordingly to the top frames.

this dynamics is close to its continuous generalization:

$$\frac{\mathrm{d}\boldsymbol{y}}{\mathrm{d}t} = f_{\boldsymbol{z}_{\lfloor t \rfloor + 1}}(\boldsymbol{y}), \tag{4.11}$$

which is a piecewise ODE. To this end, we refine this Euler approximation during testing by halving Δt; if this maintains the performance of our model, then the dynamics of the latter is close to the piecewise ODE. As shown in Figure 4.5 and Table 4.2, prediction performances overall remain stable while generating twice as many frames (cf. Appendix B.4 for further discussion). Therefore, the justification of the proposed update rule is supported by empirical evidence. This property can be used to generate videos at a higher frame rate, with the same model, and without supervision. We show in Figure 4.8 and Appendix B.4 frames generated at a double and quadruple frame rate on KTH, Human3.6M and BAIR.

4.4.3.2. Disentangling Dynamics and Content

Let us show that the proposed model actually separates content from dynamics as discussed in Section 4.3.2. To this end, two sequences $\boldsymbol{x}^{\mathrm{s}}$ and $\boldsymbol{x}^{\mathrm{t}}$ are drawn from the Human3.6M testing set. While $\boldsymbol{x}^{\mathrm{s}}$ is used for extracting our content variable $\boldsymbol{w}^{\mathrm{s}}$, dynamic states $\boldsymbol{y}^{\mathrm{t}}$ are inferred with our model from $\boldsymbol{x}^{\mathrm{t}}$. New frame sequences $\widehat{\boldsymbol{x}}$ are finally generated from the fusion of the content vector and the dynamics. This results in a content corresponding to the first sequence $\boldsymbol{x}^{\mathrm{s}}$ and a movement following the dynamics of the second sequence $\boldsymbol{x}^{\mathrm{t}}$, as observed in Figure 4.9. More samples for KTH, Human3.6M, and BAIR are presented in Appendix B.6.

4.4.3.3. Interpolation of Dynamics

Our state-space structure allows us to learn semantic representations in $\boldsymbol{y}_t$. To highlight this feature, we test whether two deterministic Moving MNIST trajectories can be

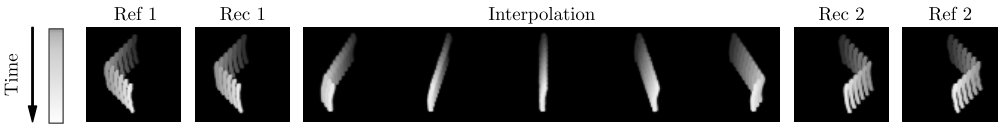


Figure 4.10.: From left to right, $\boldsymbol{x}^{\mathrm{s}}$, $\widehat{\boldsymbol{x}}^{\mathrm{s}}$ (reconstruction of $\boldsymbol{x}^{\mathrm{s}}$ by the VAE of our model), results of the interpolation in the latent space between $\boldsymbol{x}^{\mathrm{s}}$ and $\boldsymbol{x}^{\mathrm{t}}$, $\widehat{\boldsymbol{x}}^{\mathrm{t}}$ and $\boldsymbol{x}^{\mathrm{t}}$. Each trajectory is materialized in shades of grey in the frames.

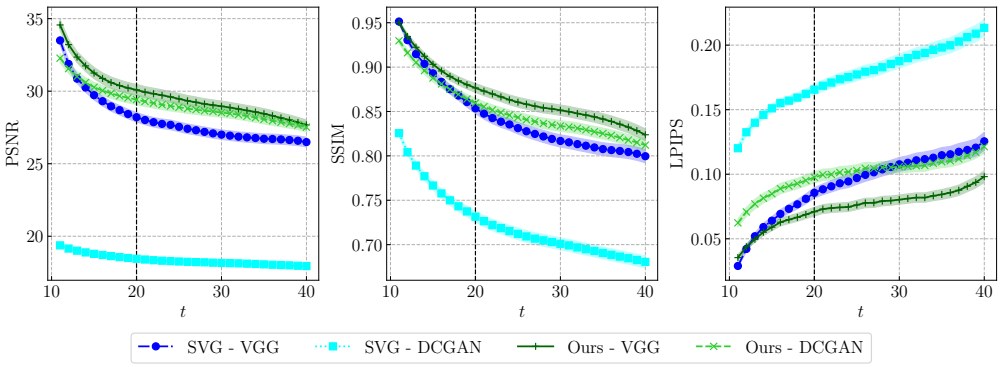


Figure 4.11.: PSNR, SSIM, and LPIPS scores with respect to t, with their 95%-confidence intervals, on the KTH dataset for SVG and our model with two choices of encoder and decoder architecture for each model: DCGAN and VGG. Dark vertical bars mark the length of training sequences.

interpolated by linearly interpolating their inferred latent initial conditions. We begin by generating two trajectories $\boldsymbol{x}^{\mathrm{s}}$ and $\boldsymbol{x}^{\mathrm{t}}$ of a single moving digit. We infer their respective latent initial conditions $\boldsymbol{y}_1^{\mathrm{s}}$ and $\boldsymbol{y}_1^{\mathrm{t}}$. We then use our model to generate frame sequences from latent initial conditions linearly interpolated between $\boldsymbol{y}_1^{\mathrm{s}}$ and $\boldsymbol{y}_1^{\mathrm{t}}$. In the case of a meaningful latent space, the resulting trajectory should also be a smooth interpolation between the directions of reference trajectories $\boldsymbol{x}^{\mathrm{s}}$ and $\boldsymbol{x}^{\mathrm{t}}$, and this is what we observe in Figure 4.10. Additional examples can be found in Appendix B.6.

4.4.3.4. Autoregressivity and Impact of the Encoder and Decoder Architecture.

Figure 4.11 and Table 4.6 expose the numerical results on KTH of our model trained with $\Delta t = 1$ and SVG (architecturally closest to ours among the considered baselines) for two choices of encoder and decoder architectures: DCGAN and VGG. We refer to Appendix B.3.1 for more details about encoder and decoder architectural choices. Since DCGAN is a less powerful architecture than VGG, results of each method with VGG are expectedly better than those of the same method with DCGAN. Moreover, our model outperforms SVG for any choice of encoder and decoder architecture, which is coherent with Figure 4.5.

Table 4.6.: FVD scores for SVG and our method on KTH, trained either with DCGAN or VGG encoders and decoders, with their 95%-confidence intervals over five different samples from the models.

SVG		Ours	
VGG	DCGAN	VGG	DCGAN
377 ± 6	542 ± 6	220 ± 2	371 ± 3

We observe, however, that the difference between a method using VGG and its DCGAN counterpart differs depending on the model. Ours shows more robustness to the choice of encoder and decoder architecture, as its performance decreases less than SVG when switching to a less powerful architecture. In this case, SVG particularly suffers in terms of PSNR, which is the metric that most penalizes dynamics errors. This shows that reducing the capacity of the encoders and decoders of SVG not only hurts its ability to produce realistic frames as expected, but also substantially lowers its ability to learn a good dynamic. We assume that this phenomenon is caused by the image-autoregressive nature of SVG, which makes it reliant on the performance of its encoders and decoders. This supports our motivation to propose a non-autoregressive model for stochastic video prediction.

4.5. Conclusion

We introduce a novel dynamic latent model for stochastic video prediction which, unlike prior image-autoregressive models, decouples frame synthesis and dynamics. This temporal model is based on residual updates of a small latent state that is shown to perform better than RNN-based models. This endows our method with several desirable properties, such as temporal efficiency and latent space interpretability. We experimentally demonstrate the performance and advantages of the proposed model, which outperforms prior state-of-the-art methods for stochastic video prediction. This work is, to the best of our knowledge, the first to propose a latent dynamic model that scales for video prediction. The proposed model is also novel with respect to the recent line of work dealing with neural networks and ODEs for temporal modeling; it is the first such residual model to scale to complex stochastic data such as videos.

We believe that the general principles of our model (state-space, residual dynamic, static content variable) can be generally applied to other models as well. Interesting future works include replacing the VRNN of Minderer et al. (2019) with our residual dynamics in order to model the evolution of key-points, supplementing our model with more video-specific priors, or leveraging its state-space nature in model-based reinforcement learning.

Chapter 5.

PDE-Driven Spatiotemporal Disentanglement

5.1. Introduction

Following the previous chapter, we delve deeper into the spatiotemporal disentanglement implemented in our video prediction model as a content variable removing information from the temporal model thanks to ad hoc inductive biases. Prediction via spatiotemporal disentanglement was first studied in video prediction works, in order to separate static and dynamic information (Denton and Birodkar, 2017) for prediction and interpretability purposes. Existing models are particularly complex, involving either adversarial losses or variational inference (see Section 2.2.2.2). Furthermore, their reliance on RNNs hinders their ability to model spatiotemporal phenomena (Yıldız, Heinonen, and Lähdesmäki, 2019; Ayed et al., 2020), as also shown in Chapter 4. Our proposition addresses these shortcomings with a simplified and improved model by grounding spatiotemporal disentanglement in the PDE formalism, in line with the growing amount of work at the interface between neural networks and dynamical systems, as discussed in Section 2.1.2.2. This allows us to generalize our results for more general spatiotemporal data beyond videos.

Spatiotemporal phenomena obey physical laws like Newton's laws of motion that lead to describe the evolution of the system through PDEs, thereby motivating our approach. Practical examples include the conservation of energy for physical systems (Hamilton, 1835), or the equation describing constant illumination in a scene (Horn and Schunck, 1981) for videos that has had a longstanding impact in computer vision with optical flow methods (Dosovitskiy, Fischer, et al., 2015; Finn, X. Y. Tan, et al., 2016). We propose to model the evolution of partially observed spatiotemporal phenomena with unknown dynamics by leveraging a formal method for the analytical resolution of PDE: the functional separation of variables (Miller, 1988). Our framework formulates spatiotemporal disentanglement for prediction as learning a separable solution, where spatial and dynamic information are represented in separate variables. Besides offering a novel interpretation of spatiotemporal disentanglement, it confers simplicity and performance compared to existing methods: disentanglement is achieved through the sole combination of a prediction objective and regularization penalties, and the temporal dynamics is defined by a learned ODE. We experimentally demonstrate the applicability, disentanglement capacity and forecasting performance of the proposed

model on various spatiotemporal phenomena involving standard physical processes and synthetic video datasets against prior state-of-the-art models.

Our work lies within the field of data-driven PDE-based spatiotemporal models, which exploit for the most part some prior physical knowledge. The latter can induce the structure of the prediction function (Brunton, Proctor, and Kutz, 2016; de Avila Belbute-Peres et al., 2018) or specific cost functions, thereby improving model performances. For instance, de Bézenac, Pajot, and Gallinari (2018) shape their prediction function with an advection-diffusion mechanism, and Z. Long, Y. Lu, X. Ma, et al. (2018) and Z. Long, Y. Lu, and Dong (2019) estimate PDEs and their solutions by learning convolutional filters proven to approximate differential operators. Samuel Greydanus, Dzamba, and Yosinski (2019), Z. Chen et al. (2020), and Toth et al. (2020) introduce non-regression losses by taking advantage of Hamiltonian mechanics (Hamilton, 1835), while Tompson et al. (2017) and Raissi, Yazdani, and Karniadakis (2020) combine physically inspired constraints and structural priors for fluid dynamic prediction. Our work deepens this literature by establishing a novel link between a resolution method for PDEs and spatiotemporal disentanglement, thereby introducing a data-agnostic model leveraging any static information in observed phenomena.

This chapter is organized as follows. Section 5.2 exposes the core principle of the separation of variables for PDEs, followed by Section 5.3 where we take inspiration from this framework to derive a spatiotemporal disentangled prediction method. Section 5.4 then evaluates the proposed model in terms of both prediction and disentanglement performance. Finally, the supplementary material referenced in this chapter is available in Appendix C.

5.2. Background: Separation of Variables

Solving high-dimensional PDEs is a difficult analytical and numerical problem (Bungartz and Griebel, 2004). Variable separation aims at simplifying it by decomposing the solution, e.g. as a simple combination of lower-dimensional functions, thus reducing the PDE to simpler differential equations.

5.2.1. Simple Case Study

Let us introduce this technique through a standard application, with proofs in Appendix C.1.1, on the one-dimensional heat diffusion problem (Fourier, 1822), consisting in a bar of length L, whose temperature at time t and position p is denoted by $u(p,t)$ and satisfies:

$$\frac{\partial u}{\partial t} = c^2 \frac{\partial^2 u}{\partial p^2}, \qquad u(0,t) = u(L,t) = 0, \qquad u(p,0) = f(p). \tag{5.1}$$

Suppose that a solution u is product-separable, i.e. it can be decomposed as:

$$u(p,t) = u_1(p) \cdot u_2(t). \tag{5.2}$$

Combined with Equation (5.1), it leads to:

$$c^2 \frac{u_1''(p)}{u_1(p)} = \frac{u_2'(t)}{u_2(t)}. \tag{5.3}$$

The left- and right-hand sides of this equation are respectively independent of t and p. Therefore, both sides are constant with respect to t and p, and solving both resulting ODEs gives solutions of the form, with $\mu \in \mathbb{R}$ and $n \in \mathbb{N}$:

$$u(p,t) = \mu \sin\left(\frac{n\pi}{L}p\right) \times \exp\left(-\left(\frac{cn\pi}{L}\right)^2 t\right). \tag{5.4}$$

The superposition principle and the uniqueness of solutions under smoothness constraints allow then to build the set of solutions of Equation (5.1) with linear combinations of separable solutions (Le Dret and Lucquin, 2016). Besides this simple example, separation of variables can be more elaborate.

5.2.2. Functional Separation of Variables

The functional separation of variables (Miller, 1988) generalizes this method. Let u be a function obeying a given arbitrary PDE with respect to time t and spatial variables s (corresponding to p in the previous heat equation example). The functional variable separation method amounts to finding a parameterization ω, a functional U, an entangling function ξ, and representations ϕ and ψ such that:

$$\omega = \xi\big(\phi(s), \psi(t)\big), \qquad\qquad u(s,t) = U(\omega). \tag{5.5}$$

With trivial choices $\xi = u$ and U, ϕ and ψ chosen as identity functions, one ensures the validity of this reformulation. Finding smarter options for ϕ, ψ, U, and ξ with regards to the initial PDE can facilitate its resolution by inducing separate simpler PDEs on ϕ, ψ, and U. For instance, product-separability is retrieved with $U = \exp$. General results on the existence of separable solutions have been proven (Miller, 1983), though their uniqueness depends on the initial conditions and the choice of functional separation (Polyanin, 2020).

Functional separation of variables finds broad applications. It helps to solve refinements of the heat equation, such as generalizations with an advection term (see Appendix C.1.2) or with complex diffusion and source terms forming a general transport equation (H. Jia et al., 2008). Besides the heat equation, functional separation of PDEs is also applicable in various physics fields like reaction-diffusion with non-linear sources or convection-diffusion phenomena (Polyanin, 2019; Polyanin and Zhurov, 2020), Hamiltonian physics (Benenti, 1997), or even general relativity (Kalnins, Miller, and G. C. Williams, 1992).

We notice that reparameterizations such as Equation (5.5) implement a separation of spatial and temporal factors of variations, i.e. spatiotemporal disentanglement. We introduce in the following a learning framework based on this general method and intuition.

5.3. Proposed Method

We propose to model spatiotemporal phenomena using the formalism of functional variable separation. We first describe our notations and then derive a principled model and constraints from this method.

5.3.1. Problem Formulation Through Separation of Variables

We consider a distribution $\mathcal{P}$ of observed spatiotemporal trajectories and corresponding observation samples $\boldsymbol{x} = (\boldsymbol{x}_{t_0}, \boldsymbol{x}_{t_0+\Delta t}, \dots, \boldsymbol{x}_{t_1})$, with $\boldsymbol{x}_t \in \mathcal{X} \subseteq \mathbb{R}^m$ and $t_1 = t_0 + \nu\Delta t$. Each sequence $\boldsymbol{x} \sim \mathcal{P}$ corresponds to an observation of a dynamical phenomenon, assumed to be described by a hidden functional $u_{\boldsymbol{x}}$ (also denoted by u for the sake of simplicity) of time $t \in \mathbb{R}$ and space coordinates s. $u_{\boldsymbol{x}}$ then characterizes the trajectory of $\boldsymbol{x}$. More precisely, $u_{\boldsymbol{x}}$ describes an unobserved continuous dynamics and $\boldsymbol{x}$ corresponds to instantaneous discrete spatial measurements associated to this dynamics. Therefore, we consider that $\boldsymbol{x}_t$ results from a time-independent function ζ of the mapping $u_{\boldsymbol{x}}(\cdot, t)$:

$$\boldsymbol{x}_t = \zeta\big(u_{\boldsymbol{x}}(\cdot, t)\big). \tag{5.6}$$

For example, $\boldsymbol{x}$ might consist in temperatures measured at some points of the sea surface, while $u_{\boldsymbol{x}}$ would be the complete ocean circulation model. In other words, $\boldsymbol{x}$ provides partial information about $u_{\boldsymbol{x}}$ and is a projection of the full dynamics. Note that this modeling entails the absence of stochasticity in the data and their evolution, in contrast with Chapter 4; this allows us to simplify the framework of analysis of spatiotemporal disentanglement and to fairly compare our model with respect to prior works on a wide range of spatiotemporal data.

We seek to learn a model which, when conditioned on prior observations, can predict future observations. To this end, we posit that the state u of each observed trajectory $\boldsymbol{x}$ is driven by a hidden PDE, shared among all trajectories; this is motivated by the extensive line of work at the crossroads of neural networks and differential equations as told in Section 2.1.2. Learning such a PDE and its solutions would then allow us to model observed trajectories $\boldsymbol{x}$. However, directly learning solutions to high-dimensional unknown PDEs is a complex task (Bungartz and Griebel, 2004; Sirignano and Spiliopoulos, 2018). We aim in this work at simplifying this resolution. We propose to do so by relying on the functional separation of variables of Equation (5.5), in order to leverage a potential separability of the hidden PDE. Therefore, analogously to Equation (5.5), we propose to formulate the problem as learning observation-constrained ϕ, ψ and U, as well as ξ and ζ, such that:

$$\omega = \xi\big(\phi(s), \psi(t)\big), \qquad u(s,t) = U(\omega), \qquad \boldsymbol{x}_t = \zeta\big(u(\cdot, t)\big), \tag{5.7}$$

with ϕ and ψ allowing to disentangle the prediction problem. In the formalism of the functional separation of variables, this amounts to decomposing the full solution u, thereby learning a spatial PDE on ϕ, a temporal ODE on ψ, and a PDE on U, as well as their respective solutions.

5.3.2. Fundamental Limits and Relaxation

Directly learning u is, however, a restrictive choice. Indeed, when formulating PDEs such as in Equation (5.1), spatial coordinates (abscissa, ordinate, depth, longitude, latitude, etc.) and time t appear as variables of the solution. Yet, unlike in fully observable phenomena studied by Sirignano and Spiliopoulos (2018) and Raissi (2018), directly accessing these variables in practice can be costly or infeasible in our partially observed setting. In other words, the nature and number of these variables are unknown. For example, the dynamics of the observed sea surface temperature is highly dependent on numerous unobserved variables such as temperature at deeper levels or wind intensity. Explicitly taking into account these unobserved variables can only be done with prior domain knowledge. To maintain the generality of the proposed approach, we choose to not make any data-specific assumption on these unknown variables.

We overcome these issues by eliminating the explicit modeling of spatial coordinates by learning dynamic and time-invariant representations accounting respectively for the time-dependent and space-dependent parts of the solution. Indeed, Equation (5.7) induces that these spatial coordinates, hence the explicit resolution of PDEs on u or U, can be ignored, as it amounts to learning ϕ, ψ and g such that:

$$\boldsymbol{x}_t = (\zeta \circ U \circ \xi)\big(\phi(\cdot), \psi(t)\big) = g\big(\phi, \psi(t)\big). \tag{5.8}$$

In order to manipulate functionals ϕ and ψ in practice, we respectively introduce learnable time-invariant and time-dependent representations of ϕ and ψ, denoted similarly to Chapter 4 by $\boldsymbol{w}$ and $\boldsymbol{y}$, such that:

$$\begin{aligned} &\phi \equiv \boldsymbol{w} \in \mathcal{S} \subseteq \mathbb{R}^d, \\ &\psi \equiv \boldsymbol{y} \colon t \mapsto \boldsymbol{y}_t \in \mathcal{T} \subseteq \mathbb{R}^p, \end{aligned} \tag{5.9}$$

where the dependence of $\psi \equiv \boldsymbol{y}$ on time t is modeled using a temporal ODE following the separation of variables, and the function ϕ, and consequently its spatial PDE, are encoded into a vectorial representation $\boldsymbol{w}$. Besides their separation of variables basis, the purpose of $\boldsymbol{w}$ and $\boldsymbol{y}$ is to capture spatial and motion information of the data. For instance, like in Chapter 4, $\boldsymbol{w}$ could encode static information such as objects appearance, while $\boldsymbol{y}$ typically contains motion variables.

$\boldsymbol{w}$ and $\boldsymbol{y}_{t_0}$, because of their dependence on $\boldsymbol{x}$ in Equations (5.8) and (5.9), are inferred from an observation history of size $\tau + 1$, i.e. $\tau + 1$ conditioning frames, $\boldsymbol{x}_{t_0::\tau}$, where $\boldsymbol{x}_{t::\tau} = (\boldsymbol{x}_t, \boldsymbol{x}_{t+\Delta t}, \ldots, \boldsymbol{x}_{t+\tau\Delta t})$, using respectively encoder networks e^S and e^T. We parameterize g of Equation (5.8) as a neural network that acts on both $\boldsymbol{w}$ and $\boldsymbol{y}_t$, and outputs the estimated observation $\widehat{\boldsymbol{x}}_t = g(\boldsymbol{w}, \boldsymbol{y}_t)$.

Unless specified otherwise, $\boldsymbol{w}$ and $\boldsymbol{y}_t$ are fed concatenated into g, which then learns the parameterization ξ of their combination. In the rest of this chapter and for the sake of simplicity, we make neural network parameterization explicit by using notations e^S_θ, e^T_θ and g_θ instead of e^S, e^T and g; θ is only the abstraction of parameterization and all introduced networks use different sets of parameters in practice, like in Chapter 4.

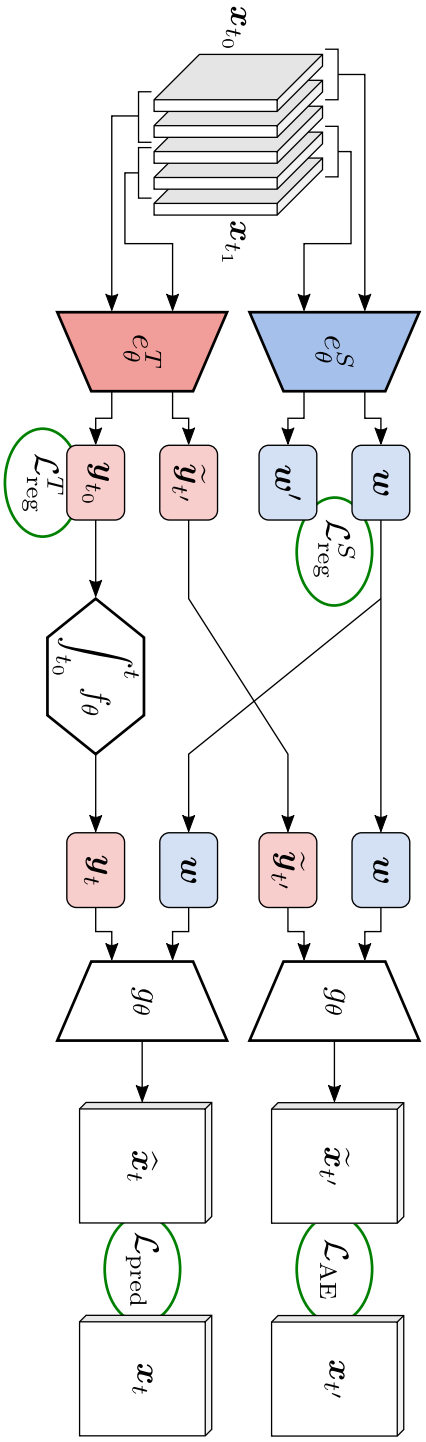


Figure 5.1.: Computational graph of the proposed model. e_θ^S and e_θ^T take contiguous observations as input; time invariance is enforced on $\boldsymbol{w}$; the evolution of $\boldsymbol{y}_t$ is modeled with an ODE and is constrained to coincide with e_θ^T; $\boldsymbol{y}_{t_0}$ is regularized; forecasting amounts to decoding from $\boldsymbol{w}$ and $\boldsymbol{y}_t$.

5.3.3. Temporal ODEs

The separation of variables allows us to partly reduce the complex task of learning and integrating PDEs to learning and integrating an ODE on ψ, which has been extensively studied in the literature, as explained in Section 2.1.2. Therefore, we model the evolution of $\boldsymbol{y}_t$, thereby the dynamics of our system, with a first-order ODE:

$$\frac{\partial \boldsymbol{y}_t}{\partial t} = f_\theta(\boldsymbol{y}_t) \qquad \Leftrightarrow \qquad \boldsymbol{y}_t = \boldsymbol{y}_{t_0} + \int_{t_0}^{t} f_\theta(\boldsymbol{y}_{t'})\,\mathrm{d}t'. \tag{5.10}$$

Following R. T. Q. Chen, Rubanova, et al. (2018), f_θ is implemented by a neural network and Equation (5.10) is solved with an ODE resolution scheme. In this work, we implement this integration using a simple Euler solving scheme with backpropagation through time (see Appendix C.5), but this could be replaced with any other solving scheme since the hereinabove formulation is general.

Suppose initial ODE conditions $\boldsymbol{w}$ and $\boldsymbol{y}_{t_0}$ have been computed with e_θ^S and e_θ^T. This leads to the following simple forecasting scheme, enforced by the corresponding regression loss:

$$\widehat{\boldsymbol{x}}_t = g_\theta\left(\boldsymbol{w}, \boldsymbol{y}_{t_0} + \int_{t_0}^{t} f_\theta(\boldsymbol{y}_{t'})\,\mathrm{d}t'\right), \quad \mathcal{L}_{\text{pred}} = \frac{1}{\nu+1}\sum_{i=0}^{\nu}\frac{1}{m}\|\widehat{\boldsymbol{x}}_{t_0+i\Delta t} - \boldsymbol{x}_{t_0+i\Delta t}\|_2^2, \tag{5.11}$$

where $\nu+1$ is the number of observations and m is the dimension of the observed variables $\boldsymbol{x}$.

Equation (5.11) ensures that the evolution of $\boldsymbol{y}$ is coherent with the observations; but we should also enforce its consistency with e_θ^T. Indeed, the dynamics of $\boldsymbol{y}_t$ is modeled by Equation (5.10), while only its initial condition $\boldsymbol{y}_{t_0}$ is computed with e_θ^T. However, there is no guaranty that $\boldsymbol{y}_t$, computed via integration, matches $e_\theta^T(\boldsymbol{x}_{t:\tau})$ at any other time t, while they should in principle coincide. We introduce the following autoencoding constraint mitigating their divergence, thereby stabilizing the evolution of $\boldsymbol{y}$:

$$\mathcal{L}_{\text{AE}} = \frac{1}{m}\left\|g_\theta\left(\boldsymbol{w}, e_\theta^T(\boldsymbol{x}_{t_0+i\Delta t:\tau})\right) - \boldsymbol{x}_{t_0+i\Delta t}\right\|_2^2, \quad \text{with } i \sim \mathcal{U}(\llbracket 0, \nu-\tau \rrbracket). \tag{5.12}$$

5.3.4. Spatiotemporal Disentanglement

As indicated hereinabove, the spatial PDE on ϕ is assumed to be encoded into $\boldsymbol{w}$. Nonetheless, since $\boldsymbol{w}$ is inferred from an observation history, we need to explicitly enforce its time independence. In the PDE formalism, this is equivalent to:

$$\frac{\partial e_\theta^S(\boldsymbol{x}_{t:\tau})}{\partial t} = 0 \qquad \Leftrightarrow \qquad \int_{t_0}^{t_1-\tau\Delta t}\left\|\frac{\partial e_\theta^S(\boldsymbol{x}_{t:\tau})}{\partial t}\right\|_2^2 \mathrm{d}t = 0. \tag{5.13}$$

However, enforcing Equation (5.13) raises two crucial issues. Firstly, in our partially observed setting, there can be variations of observable content, for instance when an object conceals another one. Therefore, strictly enforcing a null time derivative is not desirable as it prevents e_θ^S to extract accessible information that could be obfuscated in the sequence. Secondly, estimating this derivative in practice in our setting is unfeasible and costly because of the coarse temporal discretization of the data and the computational cost of e_θ^S; see Appendix C.2 for more details. We instead introduce a discretized penalty in our minimization objective, discouraging variations of content between two distant time steps, with d being the dimension of $\boldsymbol{w}$:

$$\mathcal{L}_{\text{reg}}^S = \frac{1}{d}\left\| e_\theta^S(\boldsymbol{x}_{t_0::\tau}) - e_\theta^S(\boldsymbol{x}_{t_1-\tau\Delta t::\tau}) \right\|_2^2. \tag{5.14}$$

It allows us to overcome the previously stated issues by not enforcing a strict invariance of $\boldsymbol{w}$ and removing the need to estimate any time derivative. Note that this formulation actually originates from Equation (5.13) using the Cauchy-Schwarz inequality (see Appendix C.2 for a more general derivation).

Abstracting the spatial ODE on ϕ from Equation (5.7) into a generic representation $\boldsymbol{w}$ leads, without additional constraints, to an underconstrained problem where spatiotemporal disentanglement cannot be guaranteed. Indeed, e_θ^S can be set to zero to satisfy Equation (5.14) without breaking any prior constraint, because static information is not prevented to be encoded into $\boldsymbol{y}$. Accordingly, information in $\boldsymbol{w}$ and $\boldsymbol{y}$ needs to be segmented.

Thanks to the design of our model, it suffices to ensure that $\boldsymbol{w}$ and $\boldsymbol{y}$ are disentangled at initial time t_0 for them be to be disentangled at all t. Indeed, the mutual information between two variables is preserved by invertible transformations. Equation (5.10) is an ODE and f, as a neural network, is Lipschitz-continuous, so the ODE flow $\boldsymbol{y}_t \mapsto \boldsymbol{y}_{t'}$ is invertible. Therefore, disentanglement between $\boldsymbol{w}$ and $\boldsymbol{y}_t$, characterized by a low mutual information between both variables, is preserved through time; see Appendix C.3 for a detailed discussion. We thus only constrain the information quantity in $\boldsymbol{y}_{t_0}$ by using a Gaussian prior to encourage it to exclusively contain necessary dynamic information:

$$\mathcal{L}_{\text{reg}}^T = \frac{1}{p}\|\boldsymbol{y}_{t_0}\|_2^2 = \frac{1}{p}\left\| e_\theta^T(\boldsymbol{x}_{t_0::\tau}) \right\|_2^2. \tag{5.15}$$

5.3.5. Loss Function

The minimized loss is a linear combination of Equations (5.11), (5.12), (5.14) and (5.15):

$$\mathcal{L}(\boldsymbol{x}) = \mathbb{E}_{v\sim\mathcal{P}}\Big[\lambda_{\text{pred}}\cdot\mathcal{L}_{\text{pred}} + \lambda_{\text{AE}}\cdot\mathcal{L}_{\text{AE}} + \lambda_{\text{reg}}^S\cdot\mathcal{L}_{\text{reg}}^S + \lambda_{\text{reg}}^T\cdot\mathcal{L}_{\text{reg}}^T\Big], \tag{5.16}$$

as illustrated in Figure 5.1. In the following, we conventionally set $\Delta t = 1$. Note that the presented approach could be generalized to irregularly sampled observation times thanks to the dedicated literature (Rubanova, R. T. Q. Chen, and Duvenaud, 2019), but this is out of the scope of this work.

5.3.6. Discussion of Differences with Chapter 4's Model

The model presented in this section contrasts with Chapter 4's model, denoted in the rest of this chapter as SRVP, in its exclusively regression-based training procedure, whereas their overall architectures framed in Figures 4.2 and 5.1 share their state-space nature and their separation of a content variable $\boldsymbol{w}$ from a dynamic, time-dependent, variable $\boldsymbol{y}$. These similarities actually originate from the separation of variables inspiration introduced in this chapter, thus supporting part of the spatiotemporal disentanglement view of Chapter 4. Similarly, this also explains the substantial discrepancies between their training procedures: the training of this chapter's model is grounded in PDE formalism, while SRVP's separation of content and dynamics described in Section 4.3.2 is ad hoc and only intuitively tackles the problem of spatiotemporal disentanglement. The forthcoming experiments show the advantage of this chapter's model at such disentanglement, thus justifying our approach.

However, we highlight that the training procedures of both models, while far from identical, could share some similarities when interpreting Equation (5.16)'s terms. Indeed, Equation (5.15) is similar to the KLD penalty on $\boldsymbol{y}_1$ in Equation (4.9) because it constrains the temporal model's initial condition to only contain necessary pieces of information for the dynamics. Equation (5.12), as an autoencoding loss, recalls the reconstruction term of Equations (2.22) and (4.9) with Gaussian posteriors. Finally, Equation (5.14) seeks to enforce content variable invariance like Equation (4.4)'s permutation-invariant network, additionally taking into account the potential variability of observable content in the observed sequences. We hypothesize that this analogy explains the satisfying spatiotemporal disentanglement of SRVP that is already observed in Section 4.4.3 but more rigorously evaluated in the following experiments.

5.4. Experiments

We study in this section the experimental results of our model on various spatiotemporal phenomena with physical, synthetic video and real-world datasets, which are briefly presented in this section and in more detail in Appendix C.4. We demonstrate the relevance of our model with ablation studies, and its performance by comparing it with more complex state-of-the-art models. Performances are assessed thanks to standard metrics (Denton and Fergus, 2018; Le Guen and Thome, 2020) MSE (lower is better) or its alternative PSNR (higher is better), and SSIM (higher is better); see also Section 4.4 for more information.

We use Python 3.8.1 and PyTorch 1.4.0 (Paszke et al., 2019) to implement our model. Each model is trained on an Nvidia GPU with CUDA 10.1. Training is done, like in Section 4.4, with mixed-precision training. We refer to Appendix C.6 for more experiments and prediction examples and to Appendix C.5 for complete training information.[1]

[1] Our source code is publicly released at the following URL: `https://github.com/JeremDona/spatiotemporal_variable_separation`.

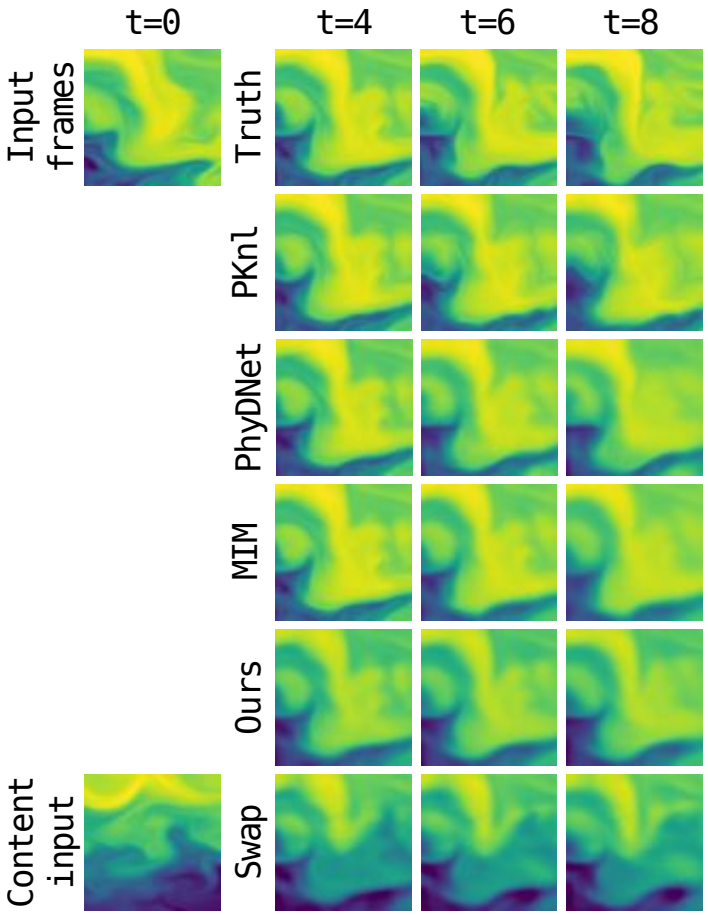


Figure 5.2.: Example of predictions of compared models on SST. Content swap preserves the location of extreme temperature regions which determine the movement while modifying the magnitude of all regions, especially in temperate areas.

Table 5.1.: Forecasting performance on WaveEq-100, WaveEq and SST of compared models with respect to indicated prediction horizons in terms of MSE and SSIM. Bold scores indicate the best performing method.

Models	WaveEq-100	WaveEq	SST			
	MSE		MSE		SSIM	
	$t+40$	$t+40$	$t+6$	$t+10$	$t+6$	$t+10$
PKnl	—	—	1.28	2.03	0.6686	0.5844
PhyDNet	—	—	1.27	1.91	0.5782	0.4645
SVG	—	—	1.51	2.06	0.6259	0.5595
MIM	—	—	0.91	1.45	0.7406	0.6525
Ours	$\mathbf{4.33 \times 10^{-5}}$	$\mathbf{1.44 \times 10^{-4}}$	**0.86**	**1.43**	**0.7466**	**0.6577**
Ours (w/o $\boldsymbol{w}$)	1.33×10^{-4}	5.09×10^{-4}	0.95	1.50	0.7204	0.6446

5.4.1. Physical Datasets: Wave Equation and Sea Surface Temperature

We first investigate two synthetic dynamical systems and a real-world dataset in order to show the advantage of PDE-driven spatiotemporal disentanglement for the forecasting of physical phenomena. To analyze our model, we first lean on the wave equation, occurring for example in acoustic or electromagnetism, with source term like Saha, Dash, and Mukhopadhyay (2020), to produce the WaveEq dataset consisting in 64×64 normalized images of the phenomenon. We additionally build the WaveEq-100 dataset by extracting 100 pixels, chosen uniformly at random and shared among sequences, from WaveEq frames; this experimental setting can be thought of as measurements from sensors partially observing the phenomenon. We also test and compare our model on the real-world dataset SST, derived from the data assimilation engine NEMO (Madec and NEMO System Team, 2019) and introduced by de Bézenac, Pajot, and Gallinari (2018), consisting in 64×64 frames showing the evolution of the sea surface temperature. Modeling its evolution is particularly challenging as its dynamic is highly non-linear, chaotic, and involves several unobserved quantities (e.g. forcing terms). Results are compiled in Table 5.1 and an example of prediction is depicted in Figure 5.2.

We compare our model on these three datasets to its alternative version with $\boldsymbol{w}$ removed and integrated into $\boldsymbol{y}$, thus also removing $\mathcal{L}^S_{\text{reg}}$ and $\mathcal{L}^T_{\text{reg}}$. We also include the state-of-the-art PhyDNet (Le Guen and Thome, 2020), MIM (Y. Wang, Jianjin Zhang, et al., 2019), SVG (Denton and Fergus, 2018) and SST-specific PKnl (de Bézenac, Pajot, and Gallinari, 2018) in the comparison on SST; only PhyDNet and PKnl were originally tested on this dataset by their authors. On the one hand, like SVG which is also investigated in Chapter 4, PhyDNet and MIM are video prediction methods; but unlike SVG, they are deterministic and operate with high-dimensional latent spaces with specific networks to model their evolution. On the other hand, PKnl incorporates physical knowledge to specifically tackle datasets of the kind of SST, by introducing an

Table 5.2.: Prediction and content swap scores of all compared models and ablations on Moving MNIST in terms of PSNR and SSIM. Bold scores indicate the best performing method.

Models	Pred. $(t+10)$		Pred. $(t+95)$		Swap $(t+10)$		Swap $(t+95)$	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
SVG	18.18	0.8329	12.85	0.6185	—	—	—	—
MIM	**24.16**	0.9113	16.50	0.6529	—	—	—	—
DrNet	14.94	0.6596	12.91	0.5379	14.12	0.6206	12.80	0.5306
DDPAE	21.17	0.8814	13.56	0.6446	**18.44**	0.8256	13.25	0.6378
PhyDNet	23.12	0.9128	16.46	0.3878	12.04	0.5572	13.49	0.2839
SRVP	22.55	**0.9201**	**18.25**	**0.8300**	17.81	0.8193	16.17	0.7736
Ours	21.70	0.9088	17.50	0.7990	18.42	**0.8368**	**16.50**	**0.7713**
- (w/o $\boldsymbol{w}$)	20.46	0.8867	14.95	0.6707	—	—	—	—
- ($\not{\mathcal{L}}_{\text{AE}}$)	21.61	0.9058	16.58	0.7611	18.21	0.8309	15.79	0.7399
- ($\not{\mathcal{L}}^{S}_{\text{reg}}$)	15.99	0.6900	12.31	0.5702	13.73	0.5476	12.07	0.5556
- ($\not{\mathcal{L}}^{T}_{\text{reg}}$)	15.63	0.7369	14.02	0.7253	14.91	0.7154	13.95	0.7234
- (GRU)	21.66	0.9088	15.45	0.4888	17.70	0.8178	14.77	0.4718

advection-diffusion mechanism in the network.

On these three datasets, our model produces more accurate long-term predictions with $\boldsymbol{w}$ than without it. This indicates that learning an invariant component facilitates training and improves generalization. The influence of $\boldsymbol{w}$ can be observed by replacing the $\boldsymbol{w}$ of a sequence with another one extracted from another sequence, changing the aspect of the prediction, as shown in Figure 5.2 (swap row, see the caption); this corresponds to the same content swap experiment performed in Section 4.4.3. We provide in Appendix C.6 further samples showing the influence of $\boldsymbol{w}$ in the prediction. Even though there is no evidence of intrinsic separability in SST, our trained algorithm takes advantage of its time-invariant component. Indeed, our model outperforms PKnl – despite the data-specific structure of the latter – as well as the stochastic SVG and the high-capacities PhyDNet and MIM models, whereas removing its static component suppresses our advantage.

We highlight that MIM is a computationally-heavy model that manipulates in an autoregressive way 64 times larger latent states than ours, hence its better reconstruction ability at the first time step. However, its sharpness and movement gradually vanish, explaining its lower performance than ours. We refer to Appendix C.6.2 for additional discussion on the application of our method and its performance on SST.

5.4.2. A Synthetic Video Dataset: Moving MNIST

We also assess the prediction and disentanglement performance of our model on the deterministic Moving MNIST dataset, like in Section 4.4. We compare our model to

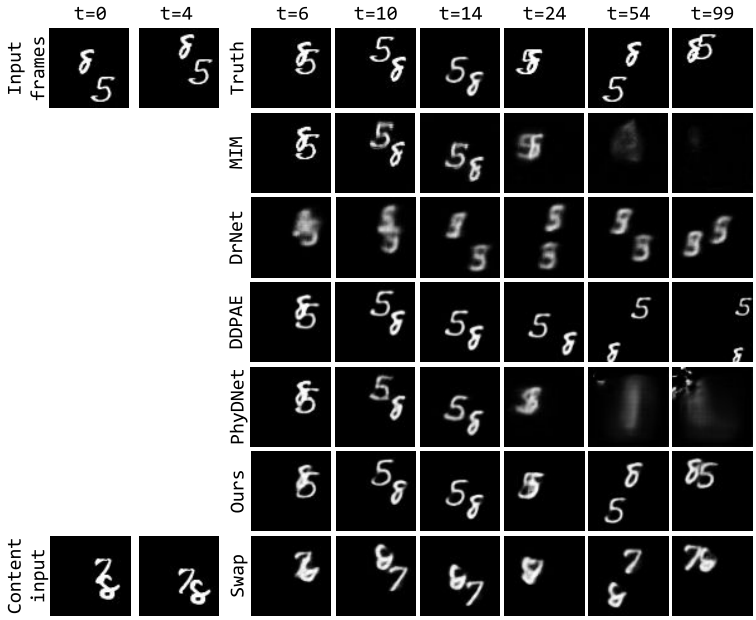


Figure 5.3.: Predictions of compared deterministic models on Moving MNIST, and content swap experiment for our model.

competitive baselines: the non-disentangled SVG and MIM, forecasting models with spatiotemporal disentanglement abilities DrNet (Denton and Birodkar, 2017), DDPAE (Hsieh et al., 2018) and PhyDNet, as well as the SRVP model previously introduced in Chapter 4. We highlight that all these models leverage powerful machine learning tools such as adversarial losses, VAEs and high-capacity temporal architectures, whereas the one proposed in this chapter is solely trained using regression penalties and small-size latent representations. We perform as well a full ablation study of our model to confirm the relevance of the introduced method.

Results reported in Table 5.2 and illustrated in Figure 5.3 correspond to two tasks: prediction and disentanglement, at both short and long-term horizons. In opposition to Section 4.4 where disentanglement is only qualitatively assessed, we quantitatively evaluate it in this chapter. To this end, we use the same task of content swapping, which consists in replacing the content representation of a sequence with the one of another sequence. In the case of a perfectly disentangled model, this should result in swapping digits of both sequences. Numerical results are then obtained by taking advantage of the synthetic nature of this dataset that allows us to implement the ground truth content swap and compare it to the generated swaps of the model.

Reported results show the advantage of our model against all baselines but SRVP. Long-term prediction challenges them as their performance and predictions collapse in the long run. This shows that the baselines, including high-capacity models MIM and PhyDNet that leverage powerful ConvLTSMs (Shi et al., 2015), have difficulties in

separating content and motion. Indeed, a model that is correctly separating content and motion should maintain digits appearance even when it miscalculates their trajectories, like DDPAE which alters only marginally the digits in Figure 5.3. In contrast, ours manages to produce consistent samples even at $t + 95$, making it reach state-of-the-art performance. Moreover, we significantly outperform all baselines in the content swap experiment, showing the clear advantage of the proposed PDE-inspired simple model for spatiotemporally disentangled prediction.

Comparison against SRVP, with the same encoder and decoder architectures, highlights the advantages of our model with its PDE inspiration. Both SRVP and this chapter's model outperform the other baselines on long-term forecasting and disentanglement, which, as noted in Section 5.3.6, can be explained by the similarities between both models. We observe as expected that SRVP outperforms our model on the forecasting task, which stems from SRVP's reliance on a powerful VAE model. However, SRVP falls short on disentanglement for both short- and long-term horizons, despite its good prediction performance which provides it with an a priori numerical advantage over our model. This signifies that, despite its simplicity, we achieve state-of-the-art disentanglement performance thanks to the PDE inspiration that allows our model to outperform SRVP, thus justifying our approach with respect to the latter.

Ablation studies also developed in Table 5.2 confirm that the advantage evidenced in the previous comparisons is due to the constraints motivated by the separation of variables. Indeed, the model without $\boldsymbol{w}$ fails at long-term forecasting, and removing any non-prediction penalty of the training loss substantially harms performances. In particular, the invariance loss on the static component and the regularization of initial condition $\boldsymbol{y}_{t_0}$ are essential, as their absence hinders both prediction and disentanglement. The auto-encoding constraint makes predictions more stable, allowing accurate long-term forecasting and disentanglement. This ablation study also confirms the necessity to constrain the ℓ_2 norm of the dynamic variable (see Equation (5.15)) for the model to disentangle. Comparisons of Table 5.2 actually show that enforcing this loss on the first time step only is sufficient to ensure state-of-the-art disentanglement, as advocated in Section 5.3.4. Finally, we assess whether the temporal ODE of Equation (5.10) induced by the separation of variables is advantageous by replacing the dynamic model with a GRU. Results reported in Table 5.2 show substantially better prediction and disentanglement performance for the original model grounded on the separation of variables, indicating the relevance of our approach.

5.4.3. A Multi-View Dataset: 3D Warehouse Chairs

We perform an additional disentanglement experiment on the 3D Warehouse Chairs dataset (Aubry et al., 2014). This dataset contains 1393 three-dimensional models of chairs seen under various angles. Since all chairs are observed from the same set of angles, this constitutes a multi-view dataset enabling quantitative disentanglement experiments. We create sequences from this dataset for our model by assembling adjacent views of each chair to simulate its rotation from right to left. We then evaluate the disentanglement properties of our model with the same content swap experiments as for Moving MNIST. It is similar to one of Denton and Birodkar (2017)'s experiments

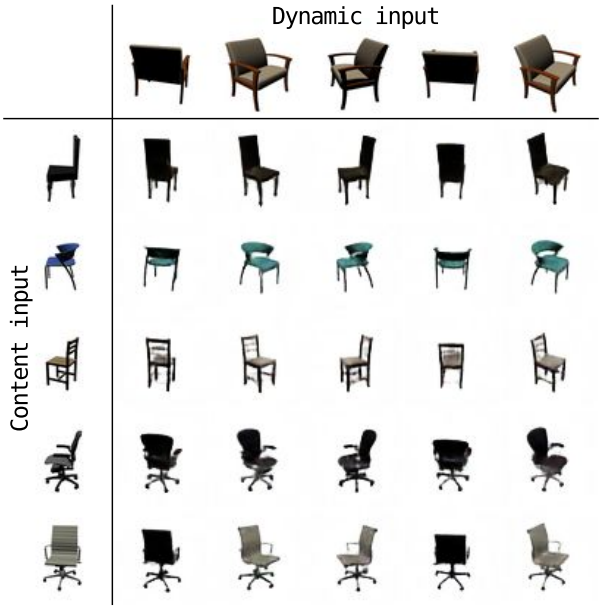


Figure 5.4.: Fusion of content (first column) and dynamic (first row) variables in our model on 3D Warehouse Chairs.

Table 5.3.: Prediction MSE ($\times 100 \times 32 \times 32 \times 2$) of compared models on TaxiBJ, with best MSE highlighted in bold.

Ours		PhyDNet	MIM	E3D	C. LSTM	PredRNN	ConvLSTM
w/ $\boldsymbol{w}$	w/o $\boldsymbol{w}$						
39.5	43.7	41.9	42.9	43.2	44.8	46.4	48.5

who qualitatively test their model on a similar, but smaller, multi-view chairs dataset. We achieve 18.70 PSNR and 0.7746 SSIM on this task, outperforming DrNet which only reaches 16.35 PSNR and 0.6992 SSIM. This is corroborated by qualitative experiments in Figures 5.4 and C.6. We highlight that the encoder and decoder architectures of both competitors are identical, validating our PDE-grounded framework for spatiotemporal disentanglement of complex three-dimensional shapes.

5.4.4. A Crowd Flow Dataset: TaxiBJ

We finally study the performance of our spatiotemporal model on the real-world TaxiBJ dataset (Junbo Zhang, Zheng, and Qi, 2017), consisting of taxi traffic flow in Beijing monitored on a 32×32 grid with an observation every thirty minutes. It is highly structured as the flows are dependent on the infrastructures of the city, and complex

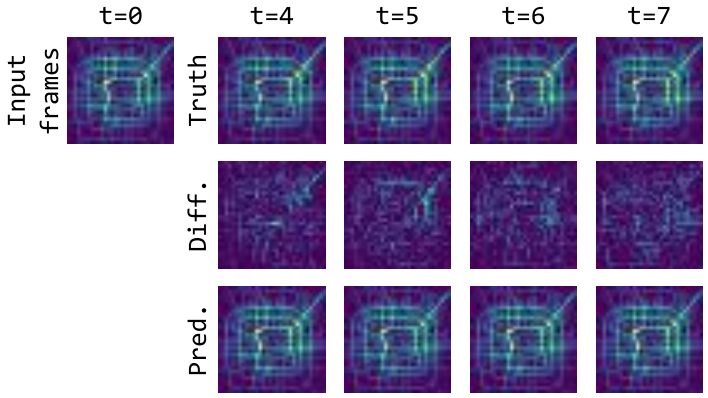


Figure 5.5.: Example of ground truth and prediction of our model on TaxiBJ. The middle row shows the scaled difference between our predictions and the ground truth.

since methods have to account for non-local dependencies and model subtle changes in the evolution of the flows. It is a standard benchmark in the spatiotemporal prediction community (Y. Wang, Jianjin Zhang, et al., 2019; Le Guen and Thome, 2020).

We compare our model in Table 5.3 against PhyDNet and MIM, as well as powerful baselines E3D-LSTM (E3D, Y. Wang, L. Jiang, et al., 2019), Causal LSTM (C. LSTM, Y. Wang, Z. Gao, et al., 2018), PredRNN (Y. Wang, M. Long, et al., 2017) and ConvLTSM (Shi et al., 2015), using results reported by Y. Wang, Jianjin Zhang, et al. (2019) and Le Guen and Thome (2020). An example of prediction is given in Figure 5.5. We observe that we significantly overcome the state of the art on this complex spatiotemporal dataset. This improvement is notably driven by the disentanglement abilities of our model, since we observe in Table 5.3 that the alternative version of our model without $\boldsymbol{w}$ achieves results comparable to E3D and worse than PhyDNet and MIM.

5.5. Conclusion

We introduce a novel method for spatiotemporal prediction inspired by the separation of variables PDE resolution technique that induces time invariance and regression penalties only. These constraints ensure the separation of spatial and temporal information. We experimentally demonstrate the benefits of the proposed model, which outperforms prior state-of-the-art methods on physical and synthetic video datasets. We believe that this work, by providing a dynamical interpretation of spatiotemporal disentanglement, could serve as the basis of more complex models further leveraging the PDE formalism. Another direction for future work could be extending the model with more involved tools such as VAEs to improve its performance, or adapt it to the prediction of natural stochastic videos like in the previous chapter.

Part IV.

Analysis of GANs' Training Dynamics

The last contribution of this thesis more generally tackles generative modeling for any data type. We are interested in theoretically and empirically analyzing the training dynamics of the most widespread deep generative model, GANs. We study the role of the discriminator in GAN training by modeling its optimization using differential equations and the theory of NTKs. We then consider the consequences of this framework on the evolution of the generated distribution during training. This contribution, detailed in the forthcoming chapter, is currently under review at an international conference and available in the following preprint.

> Jean-Yves Franceschi, Emmanuel de Bézenac, Ibrahim Ayed, Mickaël Chen, Sylvain Lamprier, and Patrick Gallinari (2021). "A Neural Tangent Kernel Perspective of GANs". In: arXiv: 2106.05566.

Chapter 6.

A Neural Tangent Kernel Perspective of GANs

6.1. Introduction

Following the steady increase of GANs applications (see Section 2.3), much effort has been put in gaining a better understanding of their training process, resulting in a vast literature on theoretical analyses of GANs. A large portion of them focus on studying GAN loss functions to conclude about their comparative advantages.

Yet, empirical evaluations (Lucic et al., 2018; Kurach et al., 2019) have shown that different GAN formulations can yield approximately the same performance in terms of sample quality and stability of the training algorithm, regardless of the chosen loss. This indicates that by focusing exclusively on the formal loss function, theoretical studies might not model practical settings adequately.

In particular, the discriminator being a trained neural network is not taken into account, nor are the corresponding inductive biases which might considerably alter the generator's loss landscape. Moreover, we show that neglecting this constraint hampers the analysis of gradient-based learning of the generator on finite training sets, since the gradient from the associated discriminator is ill-defined everywhere. These limitations thus hinder the potential of theoretical analyses to explain the empirical behavior of GANs.

In this work, we provide a framework of analysis for GANs solving these issues by explicitly incorporating the discriminator's architecture. To this end, we leverage the recent developments in the theory of deep learning driven by NTKs (Jacot, Gabriel, and Hongler, 2018), and develop theoretical results demonstrating the relevance of our approach. We first characterize the trained infinite-width discriminator under mild conditions on its architecture and loss. Then, we establish its differentiability by proving novel regularity results on its NTK. This confirms that our framework does overcome the limitations of previous analyses, making it closer to GAN practice.

This more accurate formalization enables us to derive new insights about the generator. We formulate the dynamics of the generated distribution via the generator's NTK, and discuss its consequences by linking it to gradient flows on probability spaces. We deduce that, under the IPM loss, the generated distribution minimizes its Maximum Mean Discrepancy (MMD) given by the discriminator's NTK with respect to the target distribution. Moreover, we release the Generative Adversarial Neural Tangent Kernel

ToolKit GAN(TK)^2 based on our framework, which we use to empirically validate our analysis; for example, we study the role of the ReLU activation in GAN architectures.

This chapter is organized as follows. Section 6.2 gives an overview of the literature in GAN analysis and NTKs to highlight the novelty and benefits of our contribution. Section 6.3 proceeds by pointing out the gradient issues in previous GAN analyses. The latter are resolved in our NTK-based framework introduced in Section 6.4, whose ability to apprehend GAN training is theoretically demonstrated. Sections 6.5 and 6.6 subsequently explore the theoretical and empirical consequences of these results on various GAN settings, leveraging the proposed framework and providing new insights on generative adversarial training. Finally, the supplementary material referenced in this chapter is located in Appendix D.

6.2. Related Work

GAN theory. A first line of research, started by Goodfellow, Pouget-Abadie, et al. (2014) and pursued by many others (Nowozin, Cseke, and Tomioka, 2016; Z. Zhou et al., 2019; R. Sun, Fang, and Schwing, 2020), studies the loss minimized by the generator. Assuming that the discriminator is optimal and can take arbitrary values, different families of divergences can be recovered. However, as noted by Arjovsky and Bottou (2017), these divergences should be ill-suited to GAN training, contrary to empirical evidence. Our framework addresses this discrepancy, as it properly characterizes the generator's loss and gradient.

Another line of work analyzes the dynamics and convergence of the generated distribution (Nagarajan and Kolter, 2017; Mescheder, Nowozin, and A. Geiger, 2017; Mescheder, A. Geiger, and Nowozin, 2018). As the studied dynamics are highly non-linear, this approach typically requires strong simplifying assumptions, e.g. restricting to linear neural networks or reducing datasets to a single data point. The most advanced analyses taking into account the discriminator's parameterization are specialized to specific models (Y. Bai, T. Ma, and Risteski, 2019), such as a linear one, or random feature models (S. Liu, Bousquet, and Chaudhuri, 2017; Balaji et al., 2021). In contrast to these works, our framework of analysis provides a more comprehensive modeling as we establish generally applicable results about the influence of the discriminator's architecture on the generator's dynamics.

NTKs. NTKs were introduced by Jacot, Gabriel, and Hongler (2018), who showed that a trained neural network in the infinite-width regime equates to a kernel method, thereby making the dynamics of the training algorithm tractable and amenable to theoretical study. This fundamental work has been followed by a thorough line of research generalizing and expanding its initial results (Arora, Du, W. Hu, et al., 2019; Bietti and Mairal, 2019; J. Lee, Xiao, et al., 2019; C. Liu, Libin Zhu, and Belkin, 2020; Sohl-Dickstein et al., 2020), developing means of computing NTKs (Novak et al., 2020; G. Yang, 2020), further analyzing these kernels (Z. Fan and Zhichao Wang, 2020; Bietti and Bach, 2021; Lin Chen and S. Xu, 2021), studying and leveraging them in practice (Z. Zhou et al., 2019; Arora, Du, Zhiyuan Li, et al., 2020; J. Lee, Schoenholz, et al.,

2020; Littwin, Myara, et al., 2020; Tancik et al., 2020), and more broadly exploring infinite-width networks (Littwin, Galanti, et al., 2020; G. Yang and E. J. Hu, 2021; Alemohammad et al., 2021). These prior works validate that NTKs can encapsulate the characteristics of neural network architectures, providing a solid theoretical basis to study the effect of architecture on learning problems.

Other works have studied the regularity of NTKs (Bietti and Mairal, 2019; G. Yang and Salman, 2019; Basri et al., 2020) but, as far as we know, ours is the first to state general differentiability results for NTKs and infinite-width networks. While Jacot, Gabriel, Ged, et al. (2019) sought to improve generators by investigating checkerboard artifacts in the light of NTKs and Chu, Minami, and Fukumizu (2020) introduced preliminary results in a simplified setting for the generator only, our contribution is the first to employ NTKs to theoretically study GAN training.

6.3. Limits of Previous Studies

We present in this section the usual GAN formulation and learning procedure and illustrate the limitations of prior analyses to motivate our framework.

First, we introduce some notations. Let $\Omega \subseteq \mathbb{R}^n$ be a closed convex set, $\mathcal{P}(\Omega)$ the set of probability distributions over Ω, and $L^2(\mu)$ the set of square-integrable functions from the support $\operatorname{supp}\mu$ of μ to $\mathbb{R}$ with respect to measure μ, with scalar product $\langle\cdot,\cdot\rangle_{L^2(\mu)}$. If $\Lambda \subseteq \Omega$, we write $L^2(\Lambda)$ for $L^2(\lambda)$, with λ the Lebesgue measure on Λ.

6.3.1. Generative Adversarial Networks

GAN algorithms seek to produce samples from an unknown target distribution $\beta \in \mathcal{P}(\Omega)$. To this extent, a generator function $g \in \mathcal{G}\colon \mathbb{R}^d \to \Omega$ parameterized by θ is learned to map a latent variable $\boldsymbol{z} \sim p_{\boldsymbol{z}}$ to the space of target samples such that the generated distribution α_g and β are indistinguishable for a discriminator network $f \in \mathcal{F}$ parameterized by ϑ. The generator and the discriminator are trained in an adversarial manner as they are assigned conflicting objectives.

Many GAN models consist in solving the following optimization problem, with $a, b, c\colon \mathbb{R} \to \mathbb{R}$:

$$\inf_{g\in\mathcal{G}}\left\{\mathcal{C}_{f^\star_{\alpha_g}}(\alpha_g) \triangleq \mathbb{E}_{\boldsymbol{x}\sim\alpha_g}\left[c_{f^\star_{\alpha_g}}(\boldsymbol{x})\right]\right\}, \tag{6.1}$$

where $c_f = c \circ f$, and $f^\star_{\alpha_g}$ is chosen to solve, or approximate, the following optimization problem:

$$\sup_{f\in\mathcal{F}}\left\{\mathcal{L}_{\alpha_g}(f) \triangleq \mathbb{E}_{\boldsymbol{x}\sim\alpha_g}\left[a_f(\boldsymbol{x})\right] - \mathbb{E}_{\boldsymbol{y}\sim\beta}\left[b_f(\boldsymbol{y})\right]\right\}. \tag{6.2}$$

For instance, Goodfellow, Pouget-Abadie, et al. (2014) used $a(x) = \log\big(1-\sigma(x)\big)$, $b(x) = c(x) = -\log\big(\sigma(x)\big)$; in Least Squares GAN (LSGAN) (Mao et al., 2017), $a(x) = -(x+1)^2$, $b(x) = (x-1)^2$, $c(x) = x^2$; and for IPMs (Müller, 1997) leveraged e.g. by Arjovsky, Chintala, and Bottou (2017), $a = b = c = \mathrm{id}$. Many more fall under this formulation (Nowozin, Cseke, and Tomioka, 2016; Lim and Ye, 2017).

Equation (6.1) is then solved using gradient descent on the generator's parameters, with at each step $j \in \mathbb{N}$:

$$\theta_{j+1} = \theta_j - \eta \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}} \left[\nabla_\theta g_{\theta_j}(\boldsymbol{z})^\top \nabla_{\boldsymbol{x}} c_{f^\star_{\alpha_{g_{\theta_j}}}}(\boldsymbol{x}) \Big|_{\boldsymbol{x}=g_{\theta_j}(\boldsymbol{z})} \right]. \tag{6.3}$$

This update is obtained via the chain rule from the generator's loss $\mathcal{C}_{f^\star_{\alpha_g}}(\alpha_g)$ in Equation (6.1). However, we highlight that the gradient applied in Equation (6.3) differs from $\nabla_\theta \mathcal{C}_{f^\star_{\alpha_g}}(\alpha_g)$: the terms taking into account the dependency of the optimal discriminator $f^\star_{\alpha_{g_\theta}}$ on the generator's parameters are discarded. This is because the discriminator is, in practice, considered to be independent of the generator in the usual setting of alternating optimization between the generator and the discriminator.

Since $\nabla_{\boldsymbol{x}} c_{f^\star_\alpha}(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} f^\star_\alpha(\boldsymbol{x}) \cdot c'\big(f^\star_\alpha(\boldsymbol{x})\big)$ in Equation (6.3), and as highlighted e.g. by Goodfellow, Pouget-Abadie, et al. (2014) and Arjovsky and Bottou (2017), the gradient of the discriminator plays a crucial role in the convergence of GANs. For example, if this vector field is null on the training data when $\alpha \neq \beta$, the generator's gradient is zero and convergence is impossible. For this reason, the following sections are devoted to developing a better understanding of this gradient field and its consequences when the discriminator is a neural network. In order to characterize the discriminator's gradient field, we must first study the discriminator itself.

6.3.2. On the Necessity of Modeling Discriminator Parameterization

For each GAN formulation, it is customary to elucidate which loss is implemented by Equation (6.2), often assuming that $\mathcal{F} = L^2(\Omega)$, i.e. the discriminator can take arbitrary values. Under this assumption, the original GAN yields the Jensen-Shannon divergence between α_g and β, and LSGAN a Pearson χ^2-divergence, for instance.

However, as pointed out by Arora, Ge, et al. (2017), the discriminator is trained in practice with a finite number of samples: both fake and target distributions are finite mixtures of Diracs, which we respectively denote as $\hat{\alpha}_g$ and $\hat{\beta}$. Let $\hat{\gamma}_g = \frac{1}{2}\hat{\alpha}_g + \frac{1}{2}\hat{\beta}$ be the distribution of training samples.

Assumption 1 (Finite training set)**.** *$\hat{\gamma}_g \in \mathcal{P}(\Omega)$ is a finite mixture of Diracs.*

In this setting, the Jensen-Shannon and χ^2-divergence are constant since $\hat{\alpha}_g$ and $\hat{\beta}$ generally do not have the same support. This is the theoretical reason given by Arjovsky and Bottou (2017) to introduce new losses, such as in WGAN (Arjovsky, Chintala, and Bottou, 2017). However, this is inconsistent with empirical results showing that GANs can be trained even without the latter losses.

Actually, perhaps surprisingly, in the alternating optimization setting used in practice – as described by Equation (6.3) – the constancy of $\mathcal{L}_{\hat{\alpha}_g}$, or even of $\mathcal{C}_{f^\star_{\alpha_g}}$, does not imply that $\nabla_{\boldsymbol{x}} c_{f^\star_{\alpha_g}}$ in Equation (6.3) is zero on these points. This stems from the gradient of Equation (6.3) ignoring the dependency of the optimal discriminator on the generator's

parameters: while $\nabla_\theta \mathcal{C}_{f^\star_{\alpha_g}}(\alpha_g)$ might be null, the gradient of Equation (6.3) differs and may not be zero, thereby changing the actual loss optimized by the generator. We refer to Section 6.5.2 and Appendix D.2.2 for further discussion.

Yet, in the previous theoretical frameworks where the discriminator can take arbitrary values, this gradient field is not even defined for any loss $\mathcal{L}_{\hat{\alpha}_g}$. Indeed, when the discriminator's loss $\mathcal{L}_{\hat{\alpha}_g}(f)$ is only computed on the empirical distribution $\hat{\gamma}_g$ (as it is the case for most GAN formulations), the discriminator optimization problem of Equation (6.2) never yields a unique optimal solution outside $\hat{\gamma}_g$. This is illustrated by the following straightforward result.

Proposition 1 (Ill-Posed Problem in $L^2(\Omega)$)**.** *Suppose that $\mathcal{F} = L^2(\Omega)$, $\operatorname{supp}\hat{\gamma}_g \subsetneq \Omega$. Then, for all $f, h \in \mathcal{F}$ coinciding over $\operatorname{supp}\hat{\gamma}_g$, $\mathcal{L}_{\hat{\alpha}_g}(f) = \mathcal{L}_{\hat{\alpha}_g}(h)$ and Equation (6.2) has either no or infinitely many optimal solutions in $\mathcal{F}$, all coinciding over $\operatorname{supp}\hat{\gamma}_g$.*

In particular, the set of solutions, if non-empty, contains non-differentiable discriminators as well as discriminators with null or non-informative gradients. This underspecification of the discriminator over Ω makes the gradient of the optimal discriminator in standard GAN analyses ill-defined.

This signifies that the loss alone does not impose any constraint on the values that $f_{\hat{\alpha}_g}$ takes outside $\operatorname{supp}\hat{\gamma}_g$, and more particularly that there are no constraints on the gradients. Therefore, an analysis beyond the loss function is necessary to precisely determine the learning problem of the generator defined by the discriminator.

6.4. NTK Analysis of GANs

To tackle the aforementioned issues, we notice that in practice, the inner optimization problem of Equation (6.2) is not solved exactly. Instead, using an alternating optimization procedure, a proxy neural discriminator is trained using several steps of gradient ascent for each update of the generator (Goodfellow, 2016). For a learning rate ε and a fixed generator g, this results in the following optimization process, from $i = 0$ to N:

$$\vartheta^g_{i+1} = \vartheta^g_i + \varepsilon \nabla_\vartheta \mathcal{L}_{\hat{\alpha}_g}(f_{\vartheta_i}), \qquad f^\star_{\hat{\alpha}_g} = f_{\vartheta^g_N}. \tag{6.4}$$

This parameterization and training of the discriminator as a neural network solve the underspecification of its gradient highlighted in the previous section, but this makes a theoretical analysis of its impact unattainable. We propose to facilitate this theoretical analysis thanks to the theory of NTKs, that we develop in Sections 6.4.1 to 6.4.3. We then leverage these results to analyze the dynamics of the generated distribution via the generator's NTK in Section 6.4.4.

6.4.1. Modeling Inductive Biases of the Discriminator in the Infinite-Width Limit

We study the continuous-time version of Equation (6.4):

$$\partial_t \vartheta^g_t = \nabla_\vartheta \mathcal{L}_{\hat{\alpha}_g}\left(f_{\vartheta^g_t}\right), \tag{6.5}$$

which we consider from now on in the infinite-width limit of the discriminator, making its analysis more tractable.

In the limit where the width of the hidden layers of $f_t \triangleq f_{\vartheta_t^g}$ tends to infinity, Jacot, Gabriel, and Hongler (2018) show that its so-called NTK $k_{\vartheta_t^g}$ remains constant during a gradient ascent such as Equation (6.5), i.e. there is a limiting kernel k such that:

$$\forall \tau \in \mathbb{R}_+,\ \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n,\ \forall t \in [0, \tau],\ k_{\vartheta_t^g}(\boldsymbol{x}, \boldsymbol{y}) \triangleq \partial_\vartheta f_t(\boldsymbol{x})^\top \partial_\vartheta f_t(\boldsymbol{y}) = k(\boldsymbol{x}, \boldsymbol{y}). \tag{6.6}$$

In particular, k only depends on the architecture of f and the initialization distribution of its parameters. The constancy of the NTK of f_t during gradient descent holds for many standard architectures, typically without bottleneck and ending with a linear layer (C. Liu, Libin Zhu, and Belkin, 2020), which is the case of most standard discriminators in the setting of Equation (6.2). We discuss in more detail the applicability of this approximation in Appendix D.2.1. We more particularly highlight that, under the same conditions, the discriminator's NTK remains constant over the whole GAN optimization process of Equation (6.3), and not only under a fixed generator.

Assumption 2 (Kernel)**.** *$k\colon \Omega^2 \to \mathbb{R}$ is a symmetric positive semi-definite kernel with $k \in L^2(\Omega^2)$.*

The constancy of the NTK simplifies the dynamics of training in the functional space. In order to express these dynamics, we must first introduce some preliminary definitions and assumptions.

Definition 1 (Functional gradient)**.** Whenever a functional $\mathcal{L}\colon L^2(\mu) \to \mathbb{R}$ has sufficient regularity, its gradient with respect to μ evaluated at $f \in L^2(\mu)$ is defined in the usual way as the element $\nabla^\mu \mathcal{L}(f) \in L^2(\mu)$ such that for all $\psi \in L^2(\mu)$:

$$\lim_{\varepsilon \to 0} \frac{1}{\varepsilon}\big(\mathcal{L}(f + \varepsilon\psi) - \mathcal{L}(f)\big) = \big\langle \nabla^\mu \mathcal{L}(f), \psi \big\rangle_{L^2(\mu)}. \tag{6.7}$$

Definition 2 (RKHS with respect to μ and kernel integral operator (Sriperumbudur, Gretton, et al., 2010))**.** If k follows Assumption 2 and $\mu \in \mathcal{P}(\Omega)$ is a finite mixture of Diracs, we define the Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}_k^\mu$ of k with respect to μ given by the Moore–Aronszajn theorem as the linear span of functions $k(\boldsymbol{x}, \cdot)$ for $\boldsymbol{x} \in \operatorname{supp} \mu$. Its kernel integral operator from Mercer's theorem is defined as:

$$\mathcal{T}_{k,\mu}\colon L^2(\mu) \to \mathcal{H}_k^\mu, \qquad h \mapsto \int_{\boldsymbol{x}} k(\cdot, \boldsymbol{x}) h(\boldsymbol{x}) \,\mathrm{d}\mu(\boldsymbol{x}). \tag{6.8}$$

Note that $\mathcal{T}_{k,\mu}$ generates $\mathcal{H}_k^\mu$, and elements of $\mathcal{H}_k^\mu$ are functions defined over all Ω as $\mathcal{H}_k^\mu \subseteq L^2(\Omega)$.

In the infinite-width limit, the results of Jacot, Gabriel, and Hongler (2018) imply that the discriminator f_t trained by Equation (6.5) obeys the following differential equation in-between generator updates:

$$\partial_t f_t = \mathcal{T}_{k,\hat{\gamma}_g}\Big(\nabla^{\hat{\gamma}_g} \mathcal{L}_{\hat{\alpha}}(f_t)\Big). \tag{6.9}$$

Within the alternating optimization of GANs at step j, f_0 would correspond to the previous discriminator step $f^\star_{\alpha_{g_{\theta_j}}} \triangleq f^j$, and $f^{j+1} = f_\tau$, with τ being the training time of the discriminator in-between generator updates.

In this section, we rely on this differential equation to gain a better understanding of the discriminator during training and its implications for training the generator. Firstly, under mild assumptions on the discriminator loss function, we prove that Equation (6.9) admits a unique solution for a given initial condition, thereby solving the indeterminacy issues. We then study the differentiability of neural networks in this regime, a necessary condition for trainability of GANs. These results are not specific to GANs but generalize to all neural networks trained under empirical losses of the form of Equation (6.2), e.g. any pointwise loss such as binary classification and regression. Finally, we expose the consequences of this analysis on the generated distribution's dynamics. Presented in the context of a discrete distribution $\hat{\gamma}_g$ but generalizable to broader distributions, all results are proved in Appendix D.1.

6.4.2. Existence, Uniqueness and Characterization of the Discriminator

The following is a positive result on the existence and uniqueness of the discriminator that also characterizes its general form, amenable to theoretical analysis.

Assumption 3 (Loss regularity)**.** *a and b from Equation (6.2) are differentiable with Lipschitz derivatives over $\mathbb{R}$.*

Theorem 1 (Solution of gradient descent)**.** *Under Assumptions 1 to 3, Equation (6.9) with initial value $f_0 \in L^2(\Omega)$ admits a unique solution $f_\cdot : \mathbb{R}_+ \to L^2(\Omega)$. Moreover, the following holds for all $t \in \mathbb{R}_+$:*

$$\forall t \in \mathbb{R}_+, f_t = f_0 + \int_0^t \mathcal{T}_{k,\hat{\gamma}_g}\left(\nabla^{\hat{\gamma}_g}\mathcal{L}_{\hat{\alpha}_g}(f_s)\right) \mathrm{d}s = f_0 + \mathcal{T}_{k,\hat{\gamma}_g}\left(\int_0^t \nabla^{\hat{\gamma}_g}\mathcal{L}_{\hat{\alpha}_g}(f_s)\,\mathrm{d}s\right). \tag{6.10}$$

As for any given training time t, there exists a unique $f_t \in L^2(\Omega)$, defined over all of Ω and not only the training set, the aforementioned issue in Section 6.3.2 of determining the discriminator associated to $\hat{\gamma}_g$ is now resolved. It is now possible to study the discriminator in its general form thanks to Equation (6.10). It involves two terms: the previous discriminator state $f_0 = f^j$, as well as the kernel operator of an integral. This integral is a function that is undefined outside $\operatorname{supp}\hat{\gamma}_g$, as by definition $\nabla^{\hat{\gamma}_g}\mathcal{L}_{\hat{\alpha}_g}(f_s) \in L^2(\hat{\gamma}_g)$. Fortunately, the kernel operator behaves like a smoothing operator, as it not only defines the function on all of Ω but embeds it in a highly structured space.

Corollary 1 (Training and RKHS)**.** *Under Assumptions 1 to 3, $f_t - f_0$ belongs to the RKHS $\mathcal{H}_k^{\hat{\gamma}_g}$ for all $t \in \mathbb{R}_+$.*

In our setting, this space is generated from the NTK k, which only depends on the discriminator architecture, and not on the considered loss function. This highlights the crucial role of the discriminator's implicit biases, and enables us to characterize its regularity for a given architecture.

6.4.3. Differentiability of the Discriminator and its NTK

We study in this section the smoothness, i.e. infinite differentiability, of the discriminator. It mostly relies on the differentiability of the kernel k, by Equation (6.10), which is obtained by characterizing the regularity of the corresponding conjugate kernel (J. Lee, Bahri, et al., 2018). Therefore, we prove the differentiability of the NTKs of standard architectures, and then conclude about the differentiability of f_t.

Assumption 4 (Discriminator architecture)**.** *The discriminator is a standard architecture (fully connected, convolutional or residual). The activation can be any standard activation function:* tanh*, softplus, ReLU-like, sigmoid, Gaussian, etc. A detailed assumption on activations is provided in Appendix D.1.1.*

Assumption 5 (Discriminator regularity)**.** *The activation function is smooth.*

Assumption 6 (Discriminator bias)**.** *Linear layers have non-null bias terms.*

We first prove the differentiability of the NTK under these assumptions.

Proposition 2 (Differentiability of k)**.** *Let k be the NTK of an infinite-width architecture following Assumption 4. For any $\boldsymbol{y} \in \Omega$:*

- *if Assumption 5 holds, then $k(\cdot, \boldsymbol{y})$ is smooth everywhere over Ω;*
- *if Assumption 6 holds, then $k(\cdot, \boldsymbol{y})$ is smooth almost everywhere over Ω.*

From Proposition 2, NTKs satisfy Assumption 2. We can thus use Corollary 1 and conclude about the differentiability of f_t.

Theorem 2 (Differentiability of f_t)**.** *Suppose that k is the NTK of an infinite-width network following Assumption 4. Then f_t is smooth everywhere over Ω when Assumption 5 holds, or almost everywhere when Assumption 6 holds.*

Remark 1 (Bias-free ReLU networks)**.** ReLU networks with hidden layers and no bias are not differentiable at $\mathbf{0}$. However, by introducing non-zero bias, this non-differentiability at $\mathbf{0}$ disappears in the NTK and the infinite-width discriminator. This observation explains some experimental results in Section 6.6.

Note that Bietti and Mairal (2019) state that the bias-free ReLU kernel is not Lipschitz even outside $\mathbf{0}$. However, we find this result to be incorrect. We further discuss this matter in Appendix D.2.3.

This result demonstrates that, for a wide range of GAN formulations, e.g. vanilla GAN and LSGAN, the optimized discriminator indeed admits gradients almost everywhere, making the gradient flow given to the generator well-defined in our framework. This supports our motivation to bring the theory closer to empirical evidence indicating that many GAN models do work in practice while their theoretical interpretation until now has been stating the opposite (Arjovsky and Bottou, 2017).

6.4.4. Dynamics of the Generated Distribution

The previous differentiability results allow us to study Equation (6.3), by ensuring the existence of $\nabla f^\star_{\hat{\alpha}_g}$. We consider Equation (6.3) in its continuous-time version like Equation (6.5), with training time ℓ as well as $g_\ell \triangleq g_{\theta_\ell}$ and $\alpha_\ell \triangleq \alpha_{g_\ell}$. The theory of NTKs enables us to describe the generated distribution's dynamics.

Proposition 3 (Dynamics of α_ℓ)**.** *Under Assumptions 4 and 5, Equation (6.3) is well-posed and yields in continuous-time, with k_{g_ℓ} the NTK of the generator g_ℓ:*

$$\partial_\ell g_\ell = -\mathcal{T}_{k_{g_\ell}, p_{\boldsymbol{z}}}\left(\boldsymbol{z} \mapsto \nabla_{\boldsymbol{x}}\, c_{f^\star_{\hat{\alpha}_{g_\ell}}}(\boldsymbol{x})\Big|_{\boldsymbol{x}=g_\ell(\boldsymbol{z})}\right). \tag{6.11}$$

Equivalently, the following continuity equation holds for the joint distribution $\alpha^{\boldsymbol{z}}_\ell$ of $\big(\boldsymbol{z}, g_\ell(\boldsymbol{z})\big)$ under $\boldsymbol{z} \sim p_{\boldsymbol{z}}$:

$$\partial_\ell \alpha^{\boldsymbol{z}}_\ell = -\nabla_{\boldsymbol{x}} \cdot \left(\alpha^{\boldsymbol{z}}_\ell \mathcal{T}_{k_{g_\ell}, p_{\boldsymbol{z}}}\left(\boldsymbol{z} \mapsto \nabla_{\boldsymbol{x}}\, c_{f^\star_{\hat{\alpha}_{g_\ell}}}(\boldsymbol{x})\Big|_{\boldsymbol{x}=g_\ell(\boldsymbol{z})}\right)\right), \tag{6.12}$$

where in particular α_ℓ is the marginalization of $\alpha^{\boldsymbol{z}}_\ell$ over $\boldsymbol{z} \sim p_{\boldsymbol{z}}$.

In the infinite-width limit of the generator, the generator's NTK is also constant: $k_{g_\ell} = k_g$; this is the setting that we consider to study the implications of the latter proposition. Suppose that there exists a functional $\mathscr{C}$ over $L^2(\Omega)$ such that:

$$c_{f^\star_{\hat{\alpha}}} = \partial_{\hat{\alpha}} \mathscr{C}(\hat{\alpha}). \tag{6.13}$$

Standard results in gradient flows theory (see Ambrosio, Gigli, and Savaré (2008, Chapter 10), or Arbel et al. (2019, Appendix A.3) for a summary) state that $\nabla c_{f^\star_{\hat{\alpha}}}$ is in this case the strong subdifferential of $\mathscr{C}(\hat{\alpha})$ for the Wasserstein geometry.

When $k_g(\boldsymbol{z}, \boldsymbol{z}') = \delta_{\boldsymbol{z}-\boldsymbol{z}'} I_n$ with δ a Dirac centered at $\mathbf{0}$, we have $\mathcal{T}_{k_g, p_{\boldsymbol{z}}} = \mathrm{id}$. Then, from Equation (6.12), $\alpha^{\boldsymbol{z}}_\ell$ follows the Wasserstein gradient flow with $c_{f^\star_{\hat{\alpha}}}$ as potential. This implies that $\mathscr{C}(\hat{\alpha}_\ell)$ is a decreasing function of the generator's training time ℓ. In other words, the generator g is trained to minimize $\mathscr{C}\big(\hat{\alpha}_g\big)$, which is the implicit objective induced by the discriminator.

In the general case, $\mathcal{T}_{k_g, p_{\boldsymbol{z}}}$ introduces interactions between generated particles as a consequence of the neural parameterization of the generator. Then, Equation (6.12) amounts to following the same gradient flow as before, but in a Stein geometry (Duncan, Nüsken, and Szpruch, 2019) – instead of a Wasserstein geometry – determined by the generator's integral operator, directly implying that in this case $\mathscr{C}(\hat{\alpha}_\ell)$ also decreases during training. This geometrical understanding opens interesting perspectives for theoretical analysis, e.g. we see that GAN training in this regime generalizes Stein variational gradient descent (Q. Liu and D. Wang, 2016), with the KLD minimization objective between generated and target distributions being replaced with $\mathscr{C}(\hat{\alpha})$.

Improving our understanding of Equation (6.12) is fundamental towards elucidating the open problem of the neural generator's convergence. Our study enables us to shed

light on these dynamics and highlights the necessity of pursuing the study of GANs via NTKs to obtain a more comprehensive understanding of them, which is the purpose of the rest of this chapter. In particular, the non-interacting case where $\mathcal{T}_{k_g,p_z} = \mathrm{id}$ already yields particularly useful insights, that we explore in Section 6.6. Moreover, we discuss in the following section standard GAN losses and attempt to determine the minimized functional $\mathscr{C}$ in these cases.

6.5. Fined-Grained Study for Specific Losses

Armed with the general framework of the previous section, we derive in this section more fine-grained results thanks to additional assumptions on the loss function covering standard GAN models. Proofs are detailed in Appendix D.1.

6.5.1. The IPM as an NTK MMD Minimizer

We study the case of the IPM loss, with the following remarkable discriminator expression, from which we deduce the objective minimized by the generator.

Proposition 4 (IPM discriminator)**.** *Under Assumptions 1 and 2, the solutions of Equation (6.9) for $a = b = \mathrm{id}$ are the functions of the form $f_t = f_0 + t f^*_{\hat{\alpha}_g}$, where $f^*_{\hat{\alpha}_g}$ is the unnormalized MMD witness function (Gretton, Borgwardt, M. J. Rasch, et al., 2012) with kernel k, yielding:*

$$f^*_{\hat{\alpha}_g} = \mathbb{E}_{\boldsymbol{x}\sim\hat{\alpha}_g}\big[k(\boldsymbol{x},\cdot)\big] - \mathbb{E}_{\boldsymbol{y}\sim\hat{\beta}}\big[k(\boldsymbol{y},\cdot)\big], \quad \mathcal{L}_{\hat{\alpha}_g}(f_t) = \mathcal{L}_{\hat{\alpha}_g}(f_0) + t\cdot \mathrm{MMD}^2_k\Big(\hat{\alpha}_g, \hat{\beta}\Big). \tag{6.14}$$

The latter result signifies that the direction of the gradient given to the discriminator at each of its optimization step is optimal within the RKHS of its NTK, stemming from the linearity of the IPM loss. The connection with MMD is especially interesting as it has been thoroughly studied in the literature (Muandet et al., 2017). If k is characteristic, a hypothesis discussed in Appendix D.2.5, then it defines a distance between distributions. Moreover, the statistical properties of the loss induced by the discriminator directly follow from those of the MMD: it is an unbiased estimator with a squared sample complexity that is independent of the dimension of the samples (Gretton, Borgwardt, M. Rasch, et al., 2007).

Remark 2 (Link with instance smoothing)**.** It is possible to show for IPMs that modeling the discriminator's architecture amounts to smoothing out the input distribution using the kernel integral operator $\mathcal{T}_{k,\hat{\gamma}_g}$ and can thus be seen as a generalization of the regularization technique for GANs called instance noise (C. K. Sønderby, Caballero, et al., 2017). This is discussed in further details in Appendix D.2.4.

Suppose that the discriminator is reinitialized at every step of the generator, with $f_0 = 0$ in Equation (6.9); this is possible with the initialization scheme of Yaoyu Zhang et al. (2020). Then, as $c = \mathrm{id}$ and from Proposition 4, $\nabla c_{f_{\hat{\alpha}}} = \tau \nabla f^*_{\hat{\alpha}_g}$, where τ is the

training time of the discriminator. The latter gradient constitutes the gradient flow of the squared MMD, as shown by Arbel et al. (2019) with convergence guarantees and discretization properties in the absence of a generator. This signifies that:

$$\mathscr{C}(\hat{\alpha}) = \tau \mathrm{MMD}_k^2\Big(\hat{\alpha}_g, \hat{\beta}\Big), \tag{6.15}$$

using the notations of Section 6.4.4.

Therefore, in the IPM case, the discriminator leads the generator to be trained to minimize the MMD between the empirical generated and target distributions, with respect to the NTK of the discriminator. This is the subject of study of Mroueh and Nguyen (2021), who derive convergence results about the generator trained in such conditions, considerations about the discriminator's NTK aside. This is, to the best of our knowledge, the first work considering the use of NTKs as kernels for the MMD, concurrently with Cheng and Yao Xie (2021).

Remark 3 (IPM and WGAN)**.** Along with a constraint on the set of functions, the IPM loss is involved in distances like the earth mover's distance $\mathcal{W}_1$ (Villani, 2009) – used in WGAN and StyleGAN (Karras, Laine, and Aila, 2019), and close to the hinge loss of BigGAN (Brock, Donahue, and Simonyan, 2019) –, the MMD – used in MMD GAN (C.-L. Li et al., 2017) –, the total variation, etc. In Proposition 4, we solely consider the IPM loss without additional constraint, besides having a neural discriminator. Our analysis implies that this natural constraint sufficiently constrains the discriminator in order to ensure the existence of its gradients, which we also show to be relevant given the aforementioned convergence results.

This is in contradiction with the recurrent assertion that the 1-Lipschitz constraint of WGAN (Arjovsky, Chintala, and Bottou, 2017), taken from the earth mover's distance, is necessary to remove the gradient issues of prior approaches. This is because the ill-definition of the gradients actually originates from the inadequacy of the conducted analyses, as we show in this work. Hence, while WGAN tackles the issues of previous analyses by changing the loss, we fundamentally address them with a refined framework. An analysis of WGAN, that we leave for future work, would require combining the neural discriminator and 1-Lipschitz constraints.

6.5.2. LSGAN, Convergence, and Emergence of New Divergences

The optimality of the discriminator can be proved when assuming that its loss function is well-behaved. Consider as an example the case of LSGAN, for which Equation (6.9) can be solved by slightly adapting the results from Jacot, Gabriel, and Hongler (2018) in the context of regression.

Proposition 5 (LSGAN discriminator)**.** *Under Assumptions 1 and 2, the solutions of Equation (6.9) for* $a = -(\mathrm{id}+1)^2$ *and* $b = -(\mathrm{id}-1)^2$ *are the functions defined for all* $t \in \mathbb{R}_+$ *as:*

$$f_t = \exp\big(-4t\mathcal{T}_{k,\hat{\gamma}_g}\big)\big(f_0 - \rho_g\big) + \rho_g, \qquad \rho_g = \frac{\mathrm{d}\Big(\hat{\beta} - \hat{\alpha}_g\Big)}{\mathrm{d}\Big(\hat{\beta} + \hat{\alpha}_g\Big)}. \tag{6.16}$$

In the previous result, ρ_g is the optimum of $\mathcal{L}_{\hat{\alpha}_g}$ over $L^2(\hat{\gamma}_g)$. When k is positive definite over $\hat{\gamma}_g$ (see Appendix D.2.5 for more details), f_t tends to the optimum for $\mathcal{L}_{\hat{\alpha}_g}$ as its limit is ρ_g over $\operatorname{supp}\hat{\gamma}_g$. Nonetheless, unlike the discriminator with arbitrary values of Section 6.3.2, f_∞ is defined over all Ω thanks to the integral operator $\mathcal{T}_{k,\hat{\gamma}_g}$. It is also the solution to the minimum norm interpolant problem in the RKHS (Jacot, Gabriel, and Hongler, 2018), therefore explaining why the discriminator does not overfit in scarce data regimes (see Section 6.6), and consequently has bounded gradients despite large training times, assuming its NTK is well-behaved. We also prove a more detailed generalization of this result for concave bounded losses in Appendix D.1.5, for which the same optimality conclusion holds.

Following the discussion initiated in Section 6.3.2 and applying it to the case of LSGAN using Proposition 5, similarly to the Jensen-Shannon, the resulting generator loss on discrete training data is constant when the discriminator is optimal. However, the gradients received by the generator are not necessarily null; see for instance the empirical analysis of Section 6.6. This is because the learning problem of the generator induced by the discriminator makes the generator minimize another loss $\mathscr{C}$, as explained in Section 6.4.4. This raises the question of determining $\mathscr{C}$ for LSGAN and other standard losses. Furthermore, the same problem arises for gradients obtained from incompletely trained discriminators f_t. Unlike the IPM case for which the results of Arbel et al. (2019) who leverage the theory of Ambrosio, Gigli, and Savaré (2008) have lead to a remarkable solution, this connection remains to be established for other adversarial losses. We leave this as future work.

6.6. Empirical Study with GAN(TK)^2

In this section, we present a selection of empirical results for different losses and architectures and evaluate the adequacy and practical implications of our theoretical framework in different settings. All experiments were designed and performed with the proposed Generative Adversarial Neural Tangent Kernel ToolKit GAN(TK)^2, that we publicly release[1] in the hope that the community leverages and expands it for principled GAN analyses. It is convenient to evaluate novel architectures and losses based on different visualizations and analyses. GAN(TK)^2 is implemented in Python (tested on versions 3.8.1 and 3.9.2 and on Nvidia GPUs with CUDA 10.2 and 11.2) and based on JAX (Bradbury et al., 2018) for tensor computations and Neural Tangents (Novak et al., 2020) for NTKs.

We focus in this work on particular experiments for the sake of clarity and as an illustration of the potential of analysis of our framework, but GAN(TK)^2 is a general-purpose toolkit centered around the infinite-width regime of the discriminator and could be leveraged for an even more extensive empirical analysis. We specifically consider the IPM and LSGAN losses for the discriminator since they are the two losses for which we derive in the previous section the analytic behavior of the discriminator in the infinite-width limit, but other losses can be studied as well in GAN(TK)^2. We leave a

[1] GAN(TK)^2 is available at `https://github.com/emited/gantk2`.

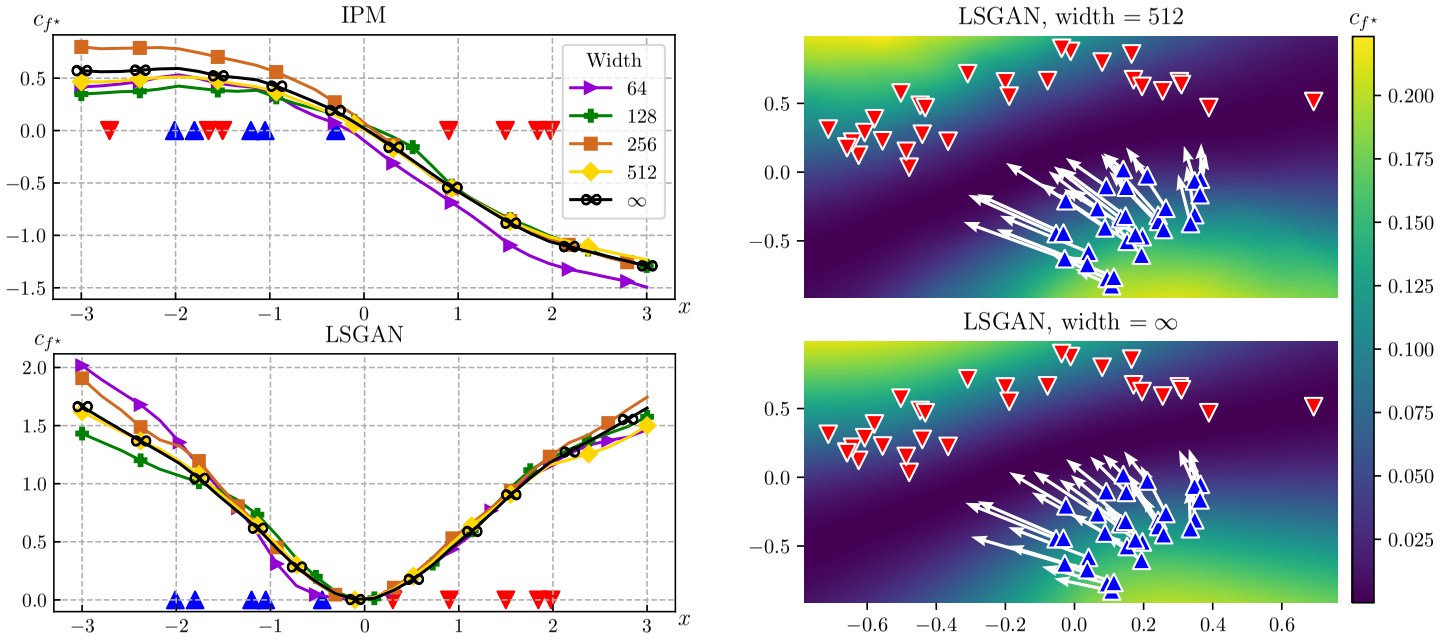


Figure 6.1.: Values of $c_{f^\star}$ for LSGAN and IPM, where $f^\star$ is a three-layer ReLU MLP with bias and varying width trained on the dataset represented by ▼ (real) and ▲ (fake) markers, initialized at $f_0 = 0$. The infinite-width network is trained for a time $\tau = 1$ and the finite-width networks using 10 gradient descent steps with learning rate $\varepsilon = 0.1$, to make training times correspond. The gradients $\nabla_{\boldsymbol{x}} c_{f^\star}$ are shown with white arrows on the two-dimensional plots for the fake distribution.

large-scale empirical study of our framework, which is out of the scope of this work, for future work.

Given the expression of the discriminator in Equation (6.10), f_0 is usually considered to be the final state of the discriminator from the previous generator update, as explained in Section 6.4.1. However, taking into account this dependency is computationally infeasible because of the analytical expressions manipulated in the infinite-width regime. Therefore, for the sake of efficiency and for these experiments only, we assume that $f_0 = 0$ – for instance, using the antisymmetrical initialization (Yaoyu Zhang et al., 2020). This also allows us to ignore residual gradients from the initialization, which can introduce some noise in the optimization process.

6.6.1. Adequacy for Fixed Distributions

Firstly, we analyze the case where generated and target distributions are fixed. In this setting, we qualitatively study the similarity between the finite- and infinite-width regimes of the discriminator and its gradients. Figure 6.1 shows $c_{f^\star}$ and its gradients on one- and two-dimensional data for LSGAN and IPM losses with a three-layer ReLU MLP with varying widths. We find the behavior of finite-width discriminators to be close to their infinite-width counterpart for commonly used widths, and converges rapidly to the given limit as the width becomes larger.

6.6.2. Convergence of Generated Distribution

We now empirically study the convergence of the generated distributions within our framework for various architectures, losses and datasets.

Experimental setting. We consider a target distribution $\hat{\beta}$ and alleviate the complexity of the analysis by following Equation (6.12) with $\mathcal{T}_{k_{g_\ell}, p_{\boldsymbol{z}}} = \mathrm{id}$, similarly to Mroueh, Sercu, and Raj (2019) and Arbel et al. (2019), thereby modeling the generator's evolution by considering a finite number of samples $\hat{\alpha}_\ell$. We study three different choices of pairs of initial distribution $\hat{\alpha}_0$ and target distribution $\hat{\beta}$:

- the 8 Gaussians problem, where $\hat{\beta}$ consists in 8 Gaussians evenly distributed on a centered sphere and $\hat{\alpha}_0$ is a standard Gaussian (see Figure 6.2, bottom left) ;
- the Density problem with more complex shapes, illustrated in Figure 6.3, bottom left;
- the AB problem, where $\hat{\alpha}_0$ and $\hat{\beta}$ are uniformly distributed in, respectively, A-shaped and B-shaped surfaces (see Figure 6.4, bottom left).

The Density and AB problems are examples taken from the Geomloss library (Feydy et al., 2019).

For both IPM and LSGAN losses, we evaluate the convergence of the generated distribution for an MLP discriminator in the finite- and infinite-width involving:

- ReLU activations, with bias terms;

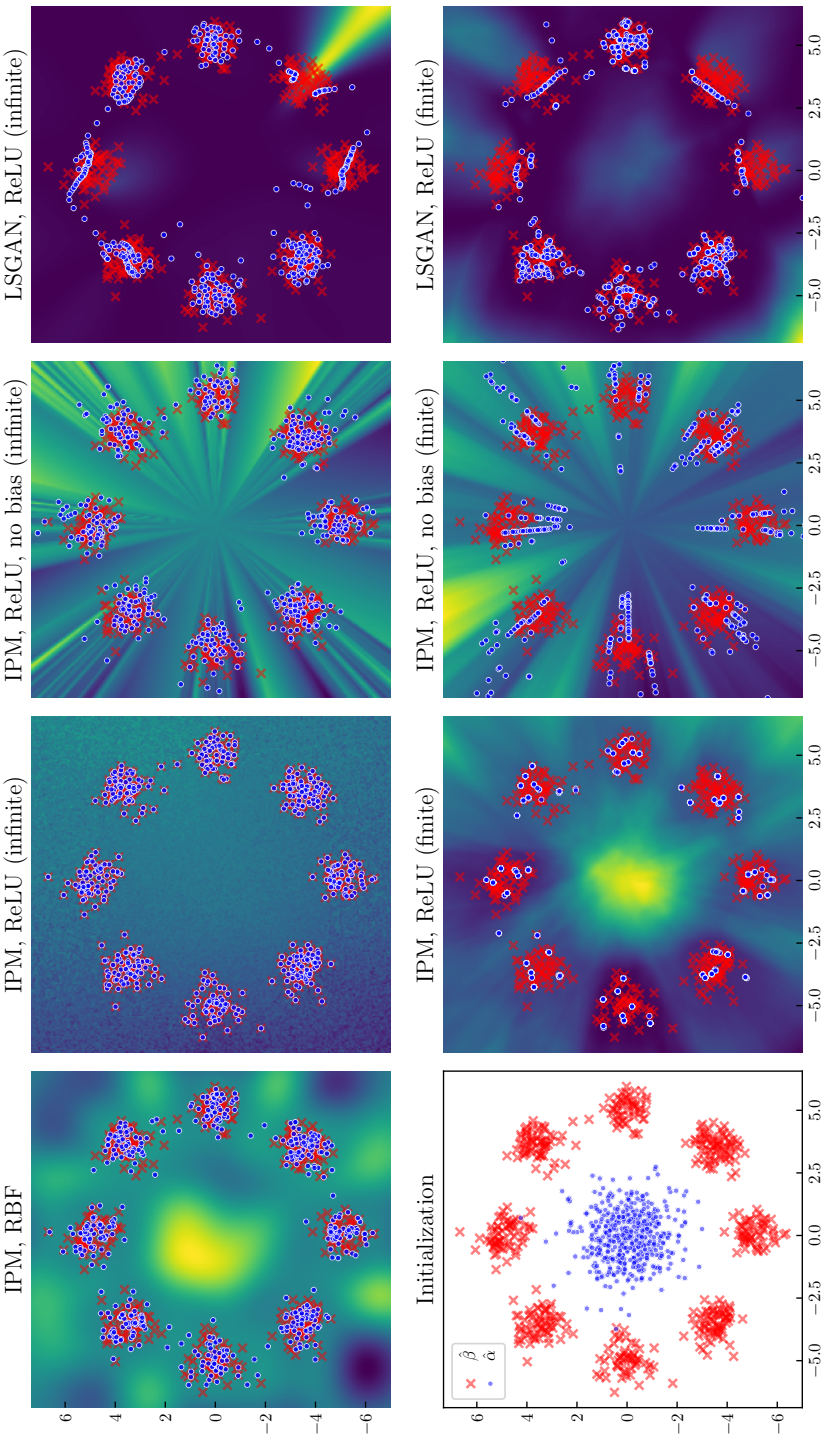


Figure 6.2.: Generator (●) and target (×) samples for different methods applied to the 8 Gaussians problem. In the background, $c_{f^\star}$.

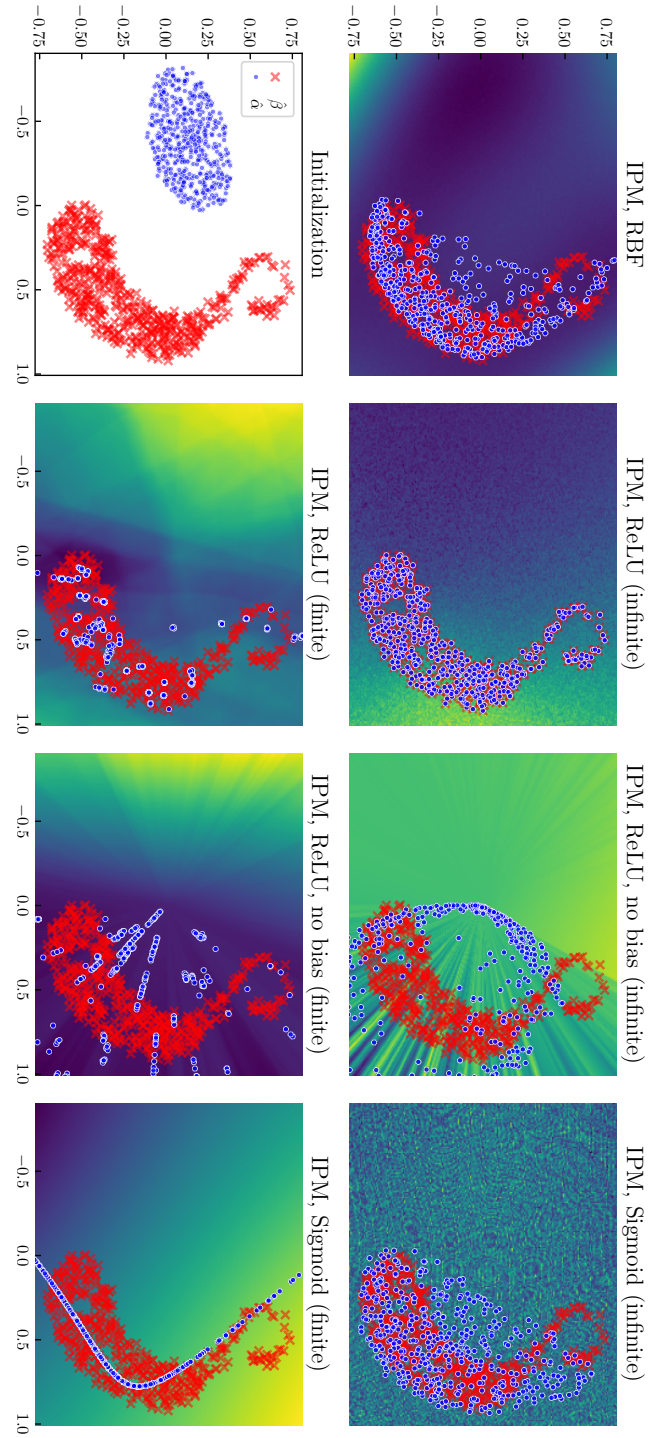


Figure 6.3.: Generator ($\bullet$) and target ($\times$) samples for different methods applied to the Density problem. In the background, $c_{f^\star}$.

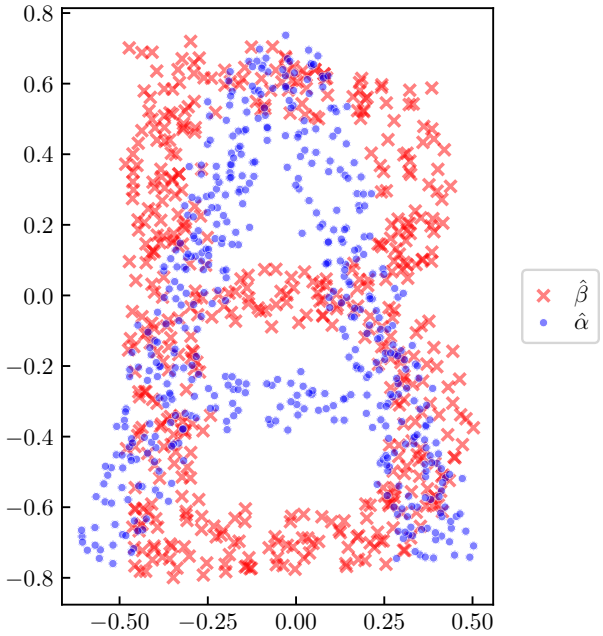


Figure 6.4.: Initial generator (●) and target (×) samples for the AB problem.

- ReLU activations, without bias terms;
- sigmoid-like activations, with bias terms.

It is also possible to comparatively evaluate the advantages of this architecture by considering the case where the infinite-width loss is not given by an NTK, but by the popular Radial Basis Function (RBF) kernel, which is characteristic and presents attractive properties (Muandet et al., 2017). Figures 6.2 and 6.3 illustrate the convergence of $\hat{\alpha}_\ell$ for some of the studied cases, and Table 6.1 compiles a complete numerical evaluation supporting our analysis.

Adequacy. We observe that performances between the finite- and infinite-width regimes are correlated, performances of ReLU networks being considerably better in the infinite-width regime. Remarkably, for the infinite-width IPM, generated and target distributions perfectly match. This can be explained by the high capacity of infinite-width neural networks and their idealized setting; it has already been shown that NTKs benefit from low-data regimes (Arora, Du, Zhiyuan Li, et al., 2020).

Impact of bias. The bias-free version of the discriminator performs worse than with bias, for both regimes and both losses. This is in line with findings of e.g. Basri et al. (2020), and can be explained in our theoretical framework by comparing their NTKs. Indeed, the NTK of a bias-free ReLU network is not characteristic, whereas its bias

Table 6.1.: Sinkhorn divergence (Feydy et al., 2019, lower is better, similar to $\mathcal{W}_2$) averaged over three runs between the final generated distribution and the target dataset for the 8 Gaussians, Density and AB problems.

Dataset	Loss	RBF kernel	ReLU	ReLU (no bias)	Sigmoid
Density	IPM (inf.)	$(2.37 \pm 0.32) \cdot 10^{-3}$	$(3.34 \pm 0.49) \cdot 10^{-9}$	$(7.34 \pm 0.34) \cdot 10^{-2}$	$(6.25 \pm 0.31) \cdot 10^{-3}$
	IPM	—	$(5.02 \pm 1.19) \cdot 10^{-3}$	$(9.25 \pm 0.30) \cdot 10^{-2}$	$(3.06 \pm 0.57) \cdot 10^{-2}$
	LSGAN (inf.)	$(7.53 \pm 0.59) \cdot 10^{-3}$	$(1.49 \pm 0.11) \cdot 10^{-3}$	$(2.80 \pm 0.03) \cdot 10^{-1}$	$(2.21 \pm 0.01) \cdot 10^{-1}$
	LSGAN	—	$(1.53 \pm 1.08) \cdot 10^{-2}$	$(1.64 \pm 0.19) \cdot 10^{-1}$	$(5.88 \pm 0.80) \cdot 10^{-2}$
AB	IPM (inf.)	$(4.65 \pm 0.82) \cdot 10^{-3}$	$(2.64 \pm 2.13) \cdot 10^{-9}$	$(6.11 \pm 0.19) \cdot 10^{-3}$	$(5.69 \pm 0.38) \cdot 10^{-3}$
	IPM	—	$(2.75 \pm 0.20) \cdot 10^{-3}$	$(3.65 \pm 1.44) \cdot 10^{-2}$	$(1.25 \pm 0.32) \cdot 10^{-2}$
	LSGAN (inf.)	$(1.13 \pm 0.05) \cdot 10^{-2}$	$(8.63 \pm 2.24) \cdot 10^{-3}$	$(1.02 \pm 0.40) \cdot 10^{-1}$	$(1.40 \pm 0.06) \cdot 10^{-2}$
	LSGAN	—	$(1.32 \pm 1.30) \cdot 10^{-1}$	$(2.57 \pm 0.73) \cdot 10^{-2}$	$(8.78 \pm 2.23) \cdot 10^{-2}$
8 Gaussians	IPM (inf.)	$(2.60 \pm 0.06) \cdot 10^{-2}$	$(9.40 \pm 2.71) \cdot 10^{-7}$	$(9.70 \pm 1.88) \cdot 10^{-2}$	$(8.40 \pm 0.02) \cdot 10^{-2}$
	IPM	—	$(1.21 \pm 0.14) \cdot 10^{-1}$	$(1.20 \pm 0.60) \cdot 10^{0}$	$(7.40 \pm 1.30) \cdot 10^{-1}$
	LSGAN (inf.)	$(4.21 \pm 0.10) \cdot 10^{-1}$	$(7.56 \pm 0.45) \cdot 10^{-2}$	$(1.27 \pm 0.01) \cdot 10^{1}$	$(7.35 \pm 0.11) \cdot 10^{0}$
	LSGAN	—	$(3.07 \pm 0.68) \cdot 10^{0}$	$(7.52 \pm 0.01) \cdot 10^{0}$	$(7.41 \pm 0.54) \cdot 10^{0}$

counterpart was proven to present powerful approximation properties (Ji, Telgarsky, and Xian, 2020). Furthermore, results of Section 6.4.3 state that the ReLU NTK with bias is differentiable at $\mathbf{0}$, whereas its bias-free version is not, which can disrupt optimization based on its gradients: note in Figure 6.2 the abrupt streaks of the discriminator directed towards $\mathbf{0}$ and their consequences on convergence.

ReLU vs. sigmoid. We observe that the sigmoid baseline is consistently outperformed by the RBF kernel and ReLU activation (with bias) for all regimes and losses. This is in accordance with common experimental practice, where internal sigmoid activations are found less effective than ReLU because of the potential activation saturation that they can induce. These results are consistent with the qualitative underperformance of sigmoid via our framework shown in Section 6.6.3.

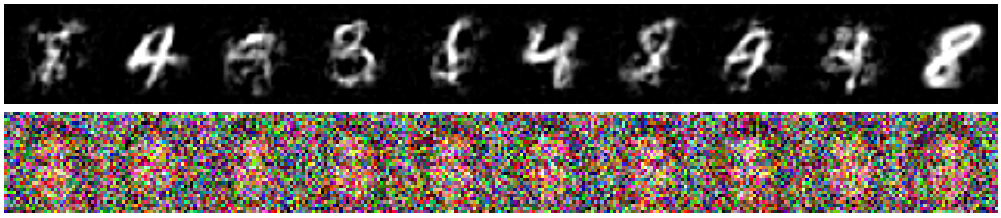
NTK vs. RBF kernel. We observe the superiority of the ReLU NTK with respect to the RBF kernel. This highlights that the gradients of a ReLU network with bias are particularly well adapted to GANs. Visualizations of the gradients given by the ReLU architecture in the infinite-width limit are available in Section 6.6.3 and further corroborate these findings. More generally, for the same reasons, we believe that the NTK of ReLU networks could be of particular interest for kernel methods requiring the computation of a spatial gradient, like Stein variational gradient descent (Q. Liu and D. Wang, 2016).

Qualitative MNIST and CelebA experiment. An experimental analysis of our framework on complex image datasets is out the scope of our study – we leave it for future work. Nonetheless, we present experiments on MNIST (LeCun, Bottou, et al., 1998) and CelebA (Z. Liu, Luo, et al., 2015) images in a similar setting as the experiments on two-dimensional point clouds of this section. For each dataset, we make a point cloud $\hat{\alpha}$, initialized to a standard Gaussian, move towards a subset of the dataset following the gradients of the IPM loss in the infinite-width regime. Qualitative results are presented in Figure 6.5.

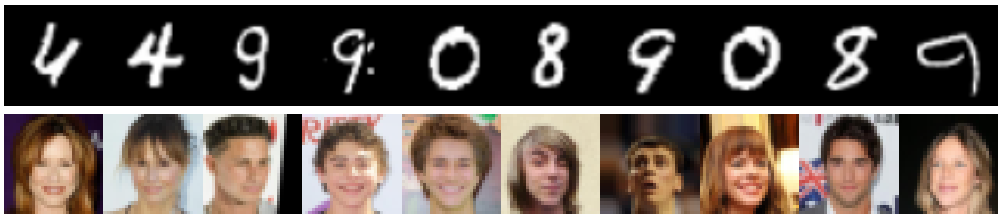
We notice, similarly to the two-dimensional experiments, that the ReLU network with bias outperforms its bias-free counterpart and a standard RBF kernel in terms of sample quality. The difference between the RBF kernel and ReLU NTK is even more flagrant in this complex high-dimensional setting, as the RBF kernel is unable to produce accurate samples.

6.6.3. Visualizing the Gradient Field Induced by the Discriminator

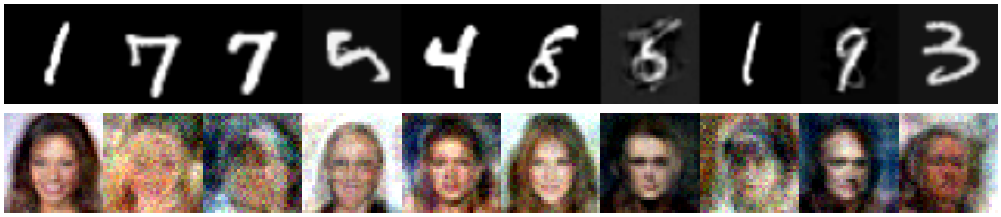
We raise in Sections 6.4.4 and 6.5 the open problem of studying the convergence of the generated distribution towards the target distribution with respect to the gradients of the discriminator. We aim in this section at qualitatively studying these gradients in a simplified case that could shed some light on the more general setting and explain some of our experimental results. These gradient fields can be plotted using the provided GAN(TK)^2 toolkit.



(a) RBF kernel: blurry digits on MNIST, prohibitively noisy images on CelebA.



(b) ReLU: sharp digits on MNIST, high-quality images on CelebA.



(c) ReLU (no bias): mostly sharp digits with some artifacts and blurry images on MNIST, blurry and noisy images on CelebA.

Figure 6.5.: Uncurated samples from the results of the descent of a set of 1024 particles over a subset of 1024 elements of MNIST and CelebA, starting from a standard Gaussian. Training is done using the IPM loss in the infinite-width kernel setting.

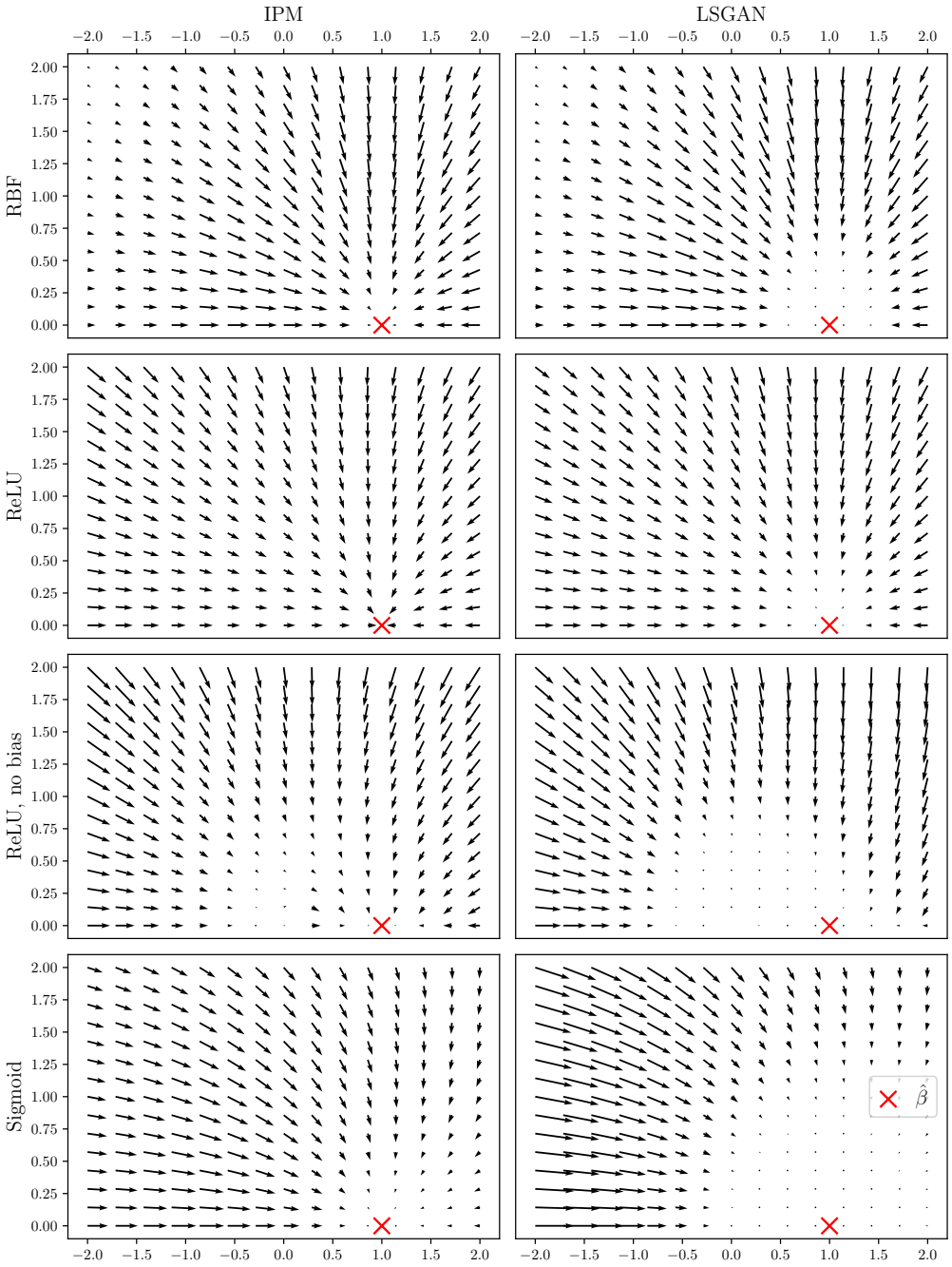


Figure 6.6.: Gradient field $\nabla c_{f^\star_{\hat{\alpha}_{\boldsymbol{x}}}}(\boldsymbol{x})$ for a generated $\boldsymbol{x} \in \mathbb{R}^2$. See Figure 6.7 for a detailed description.

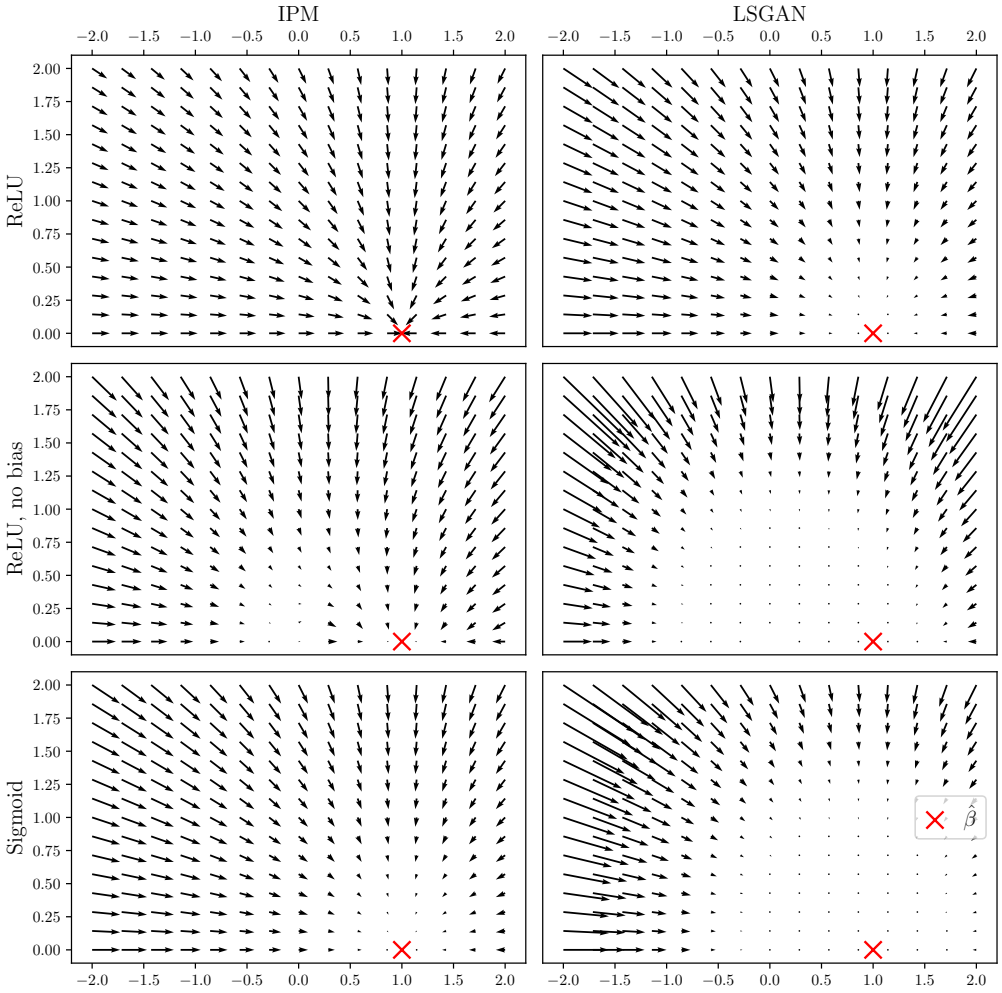


Figure 6.7.: Same plot as Figure 6.6 but with underlying points $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^{512}$. This figure and Figure 6.6 show the gradient field $\nabla c_{f^\star_{\hat{\alpha}_{\boldsymbol{x}}}}(\boldsymbol{x})$ for a generated $\boldsymbol{x}$ (i.e. $\hat{\alpha} = \delta_{\boldsymbol{x}}$) initialized to $\boldsymbol{x}_0$ with respect to its coordinates in $\mathrm{Span}\{\boldsymbol{x}_0, \boldsymbol{y}\}$ where $\boldsymbol{y}$ is the target $\times$ (i.e. $\hat{\beta} = \delta_{\boldsymbol{y}}$), with $\|\boldsymbol{y}\| = 1$. Arrows reflect the movement of $\boldsymbol{x}$ in $\mathrm{Span}\{\boldsymbol{x}_0, \boldsymbol{y}\}$ following $\nabla c_{f^\star_{\hat{\alpha}_{\boldsymbol{x}}}}(\boldsymbol{x})$; scales are specific for each pair of loss and network. The ideal case is the convergence of $\boldsymbol{x}$ along this gradient field towards the target $\boldsymbol{y}$. Note that in the chosen orthonormal coordinate system from $\mathrm{Span}\{\boldsymbol{x}_0, \boldsymbol{y}\}$, without loss of generality, $\boldsymbol{y}$ has coordinate $(1, 0)$; moreover, the gradient field is symmetrical with respect to the horizontal axis.

6.6.3.1. Setting

By Theorem 1, for any loss and any training time, the discriminator can be expressed as $f^\star_{\hat{\alpha}_g} = \mathcal{T}_{k,\hat{\gamma}_g}(h_0)$, for some $h_0 \in L^2(\hat{\gamma}_g)$. Thus, there exists $h_1 \in L^2(\hat{\gamma}_g)$ such that:

$$f^\star_{\hat{\alpha}_g} = \sum_{\boldsymbol{x} \in \operatorname{supp} \hat{\gamma}_g} h_1(\boldsymbol{x}) k(\boldsymbol{x}, \cdot). \tag{6.17}$$

Consequently:

$$\nabla f^\star_{\hat{\alpha}_g} = \sum_{\boldsymbol{x} \in \operatorname{supp} \hat{\gamma}_g} h_1(\boldsymbol{x}) \nabla k(\boldsymbol{x}, \cdot), \quad \nabla c_{f^\star_{\hat{\alpha}_g}} = \sum_{\boldsymbol{x} \in \operatorname{supp} \hat{\gamma}_g} h_1(\boldsymbol{x}) \nabla k(\boldsymbol{x}, \cdot) c'\Big(f^\star_{\hat{\alpha}_g}(\cdot)\Big). \tag{6.18}$$

Dirac GAN setting. The latter linear combination of gradients indicates that, by examining gradients of $c_{f^\star_{\hat{\alpha}_g}}$ for pairs of $(\boldsymbol{x}, \boldsymbol{y}) \in \operatorname{supp} \hat{\alpha}_g \times \operatorname{supp} \hat{\beta}$, one could already develop potentially valid intuitions that can hold even when multiple points are considered. This is especially the case for the IPM loss, as h_0, h_1 have a simple form: $h_1(\boldsymbol{x}) = 1$ if $\boldsymbol{x} \in \operatorname{supp} \hat{\alpha}_g$ and $h_1(\boldsymbol{y}) = -1$ if $\boldsymbol{y} \in \operatorname{supp} \hat{\alpha}_g$ (assuming points from $\hat{\alpha}_g$ and $\hat{\beta}_g$ are uniformly weighted); moreover, note that $c'\Big(f^\star_{\hat{\alpha}_g}(\cdot)\Big) = 1$. Thus, we study here $\nabla c_{f^\star_{\hat{\alpha}_g}}$ when $\hat{\alpha}_g$ and $\hat{\beta}_g$ are only comprised of one point, i.e. the setting of Dirac GAN (Mescheder, A. Geiger, and Nowozin, 2018), with $\hat{\alpha}_g = \delta_{\boldsymbol{x}} \triangleq \hat{\alpha}_{\boldsymbol{x}}$ and $\hat{\beta}_g = \delta_{\boldsymbol{y}}$.

Visualizing high-dimensional inputs. Unfortunately, the gradient field is difficult to visualize when the samples live in a high-dimensional space. Interestingly, the NTK $k(\boldsymbol{x}, \boldsymbol{y})$ for any architecture starting with a fully connected layer only depends on $\|\boldsymbol{x}\|$, $\|\boldsymbol{y}\|$ and $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ (G. Yang and Salman, 2019), and therefore all the information of $\nabla c_{f^\star_{\hat{\alpha}_{\boldsymbol{x}}}}$ is contained in $\operatorname{Span}\{\boldsymbol{x}, \boldsymbol{y}\}$. From this, we show in Figures 6.6 and 6.7 the gradient field $\nabla c_{f^\star_{\hat{\alpha}_{\boldsymbol{x}}}}(\boldsymbol{x})$ in the two-dimensional space $\operatorname{Span}\{\boldsymbol{x}, \boldsymbol{y}\}$ for different architectures and losses in the infinite-width regime described in Section 6.6 and in this section. Figure 6.6 corresponds to two-dimensional $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^2$, and Figure 6.7 to high-dimensional $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^{512}$. Note that in the plots, the gradient field is symmetric with respect to the horizontal axis. For this reason, we restrict the illustration to the case where the second coordinate is positive.

Convergence of the gradient flow. In the last paragraph, we have seen that the gradient field in the Dirac-GAN setting lives in the two-dimensional $\operatorname{Span}\{\boldsymbol{x}, \boldsymbol{y}\}$, independently of the dimensionality of $\boldsymbol{x}, \boldsymbol{y}$. This means that when training the generated distribution, the position of the particle $\boldsymbol{x}$ always remains in this two-dimensional space, and hence convergence in this setting can be easily checked by studying this gradient field. This is what we do in the following, for different architectures and losses.

6.6.3.2. Qualitative Analysis of the Gradient Field

$\boldsymbol{x}$ is far from $\boldsymbol{y}$. When generated outputs are far away from the target, it is essential that their gradient has a large enough magnitude in order to pull these points towards

the target. The behavior of the gradients for distant points can be observed in the plots. For ReLU networks, for both losses, the gradients for distant points seem to be well behaved and large enough. Note that in the IPM case, the magnitude of the gradients is even larger when $\boldsymbol{x}$ is further away from $\boldsymbol{y}$. This is not the case for the RBF kernel when the variance parameter is too small, as the magnitude of the gradient becomes prohibitively small. We highlight that we select a large variance parameter in order to avoid such a behavior, but diminishing magnitudes can still be observed. Note that choosing an overly large variance may also have a negative impact on the points that are closer to the target.

$\boldsymbol{x}$ is close to $\boldsymbol{y}$. A particularity of the NTK of ReLU discriminators with bias that arises from this study is that the gradients vanish more slowly when the generated $\boldsymbol{x}$ tends to the target $\boldsymbol{y}$, compared to NTKs of ReLU without bias and sigmoid networks, and to the RBF kernel. We hypothesize that this is another distinguishing feature that helps the generated distribution to converge more easily to the target distribution, especially when they are not far apart. On the contrary, this gradient vanishes more rapidly for NTKs of ReLU without bias and sigmoid networks, compared to the RBF kernel. This can explain the worse performance of such NTKs compared to the RBF kernel in our experiments (see Table 6.1). Note that this phenomenon is even more pronounced in high-dimensional spaces such as in Figure 6.7.

$\boldsymbol{x}$ is close to 0. Finally, we highlight gradient vanishing and instabilities around the origin for ReLU networks without bias. This is related to its differentiability issues at the origin exposed in Section 6.4.3, and to its lack of representational power discussed in Appendix D.2.5. This can also be retrieved on larger scale experiments of Figures 6.2 and 6.3 where the origin is the source of instabilities in the descent.

Sigmoid network. Figures 6.2 and 6.3 provide a clear explanation of the aforementioned underperformance of the sigmoid activation: as stated above, the magnitudes of the gradients become too small when $\boldsymbol{x} \to \boldsymbol{y}$, and heavily depend on the direction from which $\boldsymbol{x}$ approaches $\boldsymbol{y}$. Ideally, the induced gradient flow should be insensitive to the direction in order for the convergence to be reliable and robust, which seems to be the case for ReLU networks.

6.7. Conclusion and Discussion

Leveraging the theory of infinite-width neural networks, we propose a framework of analysis of GANs explicitly modeling a large variety of discriminator architectures. We show that the proposed framework models more accurately GAN training compared to prior approaches by deriving properties of the trained discriminator. We demonstrate the analysis opportunities of the proposed modeling by further studying the generated distribution for specific GAN losses and architectures, both theoretically and empirically, notably using our public GAN analysis toolkit. We believe that this work will serve as a basis for more elaborate analyses, thus leading to more principled, better GAN models.

Part V.

Conclusion

We conclude this document with an overview of the work performed during this thesis in Chapter 7, and an opening in Chapter 8 describing potential future work.

Chapter 7.

Overview of our Work

This thesis started in September 2018. The previous chapters cover the work produced during this time on this topic. In conclusion, we summarize our contributions in Section 7.1 and then highlight in the rest of this chapter some work and context that was not covered in the previous chapters. More precisely, we emphasize in Section 7.2 our efforts at ensuring the reproducibility of our research, acknowledge in Section 7.3 the people who participated in producing the presented contributions as well as related fundings, and succinctly expose in Section 7.4 other work performed during this thesis but not directly related to the topics covered by this document.

7.1. Summary of Contributions

We briefly summarize in this section the contributions of this thesis; a more detailed summary per chapter is available in Section 1.2.

During this thesis, we have worked on multiple aspects of temporality in neural networks through the prism of representation learning and generative modeling, with a focus on dynamical systems and differential equations.

This has led us to first design a general-purpose unsupervised representation learning method for time series, aiming at embedding them into a fixed-size representation in a scalable way via a time-based triplet loss. We have then tackled representation learning for temporal data by learning a low-dimensional state per time step of a sequence and in-between transitions in a state-space manner. This equivalence between learning to represent and learning to predict has motivated the introduction of prediction models based on state-space representations. We have highlighted, to this end, the crucial role of designing appropriate transition functions between representations, which we have shown to be most useful when obeying differential equations parameterized by neural networks. We have applied these generative methods to spatiotemporal data like videos and physical phenomena by considering the challenges and possibilities they introduce, such as stochastic forecasting and spatiotemporal disentanglement. Finally, we have more broadly addressed generative modeling for all kinds of data by theoretically and empirically studying the training dynamics of the state-of-the-art generative models, GANs. In particular, we have described the evolution of the neural network discriminator of GANs with respect to its training time with a differential equation thanks to the theory of NTKs, and its consequences on the generated distribution's flow throughout training. Such modeling of the architecture and optimization of the

discriminator have allowed us to better apprehend the training of GANs, thus showing the benefits of studying neural networks as dynamical systems even when the data are not temporal.

7.2. Reproducibility

Throughout this document, we mention our efforts at producing reproducible research. The machine learning community has increasingly addressed this issue since the beginning of this thesis (Pineau et al., 2021), especially because of the explosion of empirical advances in the field. We have tried to follow this necessary evolution for each of our contributions, answering positively to all reproducibility requirements of our articles' submissions to international conferences.

More particularly, we have taken the following actions in this direction:

- algorithms and theoretical results have been precisely described and proved when necessary;
- we have detailed and made available whenever possible the considered datasets, in both this document and the original articles, as well as in the released source code;
- the source code of all proposed models, frameworks and experiments has been publicly released under a free license and with detailed instructions;
- we have made sure to answer any request and question about our code in their Git repositories, pre- and post-publication;
- all infrastructures and hyperparameters for our experiments have been specified, in both this document and the original articles, as well as in the released source code with easy-to-launch commands;
- when possible with the available computational resources, we have complemented our numerical results with error bars and variance;
- we have explicated how we have produced the results of all the considered baselines in our experiments.

7.3. Acknowledgements

We would like to highlight that the presented contributions have been made in collaboration with other Ph.D. students and researchers who share their paternity, in order of appearance: Aymeric Dieuleveut, Martin Jaggi, Edouard Delasalles, Mickaël Chen, Sylvain Lamprier, Patrick Gallinari, Jérémie Donà, Emmanuel de Bézenac and Ibrahim Ayed. We also emphasize the importance of fruitful discussion and feedback from other researchers that have helped us to improve our articles, including the anonymous reviewers of our papers, our peers Sidak Pal Singh, Andreas Hug, Jean-Baptiste

Cordonnier, Andreas Loukas, François Fleuret, Matthias Minderer and Vincent Le Guen, as well as all members of the MLIA team of the LIP6 laboratory within the time frame of this thesis.

Moreover, we acknowledge the fundings that have supported this thesis and the produced work. First of all, the École Normale Supérieure de Lyon, Sorbonne Université and the Ministère de l'Enseignement Supérieur, de la Recherche et de l'Innovation have funded this thesis. This work was granted access to the HPC resources of IDRIS under allocations 2020-AD011011360 and 2021-AD011011360R1 made by GENCI (Grand Équipement National de Calcul Intensif), which have been crucial to the research presented in Chapters 4 to 6. The study of Chapter 5 has been conducted using E.U. Copernicus Marine Service Information. Finally, the work of this thesis has benefited from the SFA-AM ETH Board initiative, the LOCUST ANR project (ANR-15-CE23-0027), CLEAR (Center for LEArning and data Retrieval, joint laboratory with Thales), the European Union's Horizon 2020 research and innovation programme under grant agreement 825619 (AI4EU), and the 2019 ANR AI Chairs program via the DL4CLIM project.

7.4. Other Works

We would finally like to succinctly mention the work performed during this thesis but that could not be included in the previous chapters.

Beyond producing research papers, this thesis has been the opportunity to peer-review other research papers, present and discuss our or other's work at various conferences and seminars, partially manage a reading group on machine learning for spatiotemporal data, and fulfill the role of teaching assistant at Sorbonne Université for undergraduate students in Computer Science.

Furthermore, we have thought about, and sometimes worked on, follow-ups of the presented contributions and possible future works. We deal with this unreleased content and perspectives in the next and last chapter of this document.

Chapter 8.

Perspectives

We expose in this chapter perspectives for future work based on the contributions presented in the previous chapters. We first present in Section 8.1 some ideas that have been worked on during this thesis but have not resulted in significant advances yet, because of a lack of time or results. We then list in Section 8.2 other possible research directions that are yet to explore.

8.1. Unfinished Projects

8.1.1. Adaptive Stochasticity for Video Prediction

Deep Markov models and VRNN, seen in Section 2.3.2.3, as well as the SRVP model of Chapter 4, are all able to perform stochastic predictions, but fundamental differences in their design distinguish them and impact their performance. Nonetheless, they share a common modeling choice: stochasticity only intervenes once per observation $\boldsymbol{x}_t$, and both are synchronized; in the continuous-time view of SRVP in Equation (4.11), sampling a new $\boldsymbol{z}$ occurs at regular time stamps. One may consider multiplying the number of $\boldsymbol{z}$ between each observation to synchronize the sampling steps with the Euler step size, but this would maintain the synchronization with the observations, although with a different frequency.

We believe that such synchronization raises modeling issues. Indeed, sampling a $\boldsymbol{z}$ variable represents a possible change of dynamics, and there is no reason for this change to occur synchronously with the observation rate. Let us take as an example the Moving MNIST dataset from the experiments of Chapters 4 and 5. Bounces may occur between observations, and not necessarily at the precise moment of the observation. Yet, all latent variables models consider this change of movement only at the moment of the observation, making them unable to assimilate the true dynamics of the dataset.

The continuous-time view of SRVP in Equation (4.11) provides, in opposition to previous models, an opportunity to tackle this issue. This is because it allows us to draw variables $(\boldsymbol{z}_{i+1})_i$ at learned time stamps $(t_i)_i$ rather than at integer ones like in SRVP, with the state $\boldsymbol{y}_t$ following an ODE in-between each pair of consecutive time stamps:

$$\frac{\mathrm{d}\boldsymbol{y}}{\mathrm{d}t} = f_\theta(\boldsymbol{y}_t, \boldsymbol{z}_{i+1}) \text{ for } t \in [t_i, t_{i+1}]. \tag{8.1}$$

Learning the time stamps would make the model decide when a change of dynamics

occurs, potentially improving its modeling and prediction abilities. Besides, its potential ability to automatically detect these changes of dynamics and possibly skip unnecessary updates may be beneficial for interpretability and computational efficiency purposes.

We have more particularly investigated an instance of such a general principle where, given previous states $\boldsymbol{y}_{\leq t_i}$ and variables $\boldsymbol{z}_{\leq i+1}$, the next time stamp is computed by determining the delay until the new dynamics based on the current state and variable:

$$\tau_i = d_\vartheta(\boldsymbol{y}_{t_i}, \boldsymbol{z}_{i+1}) > 0, \qquad t_{i+1} = t_i + \tau_i, \tag{8.2}$$

where d_ϑ is a ϑ-parameterized neural network with positive outputs (e.g. with a final softplus activation). Learning delays enables the temporal model to work in relative rather than absolute time, which facilitates its specification.

A practical issue of such a model for batch training is that all sequences of a batch are desynchronized, with different delays and time stamps. This forbids efficient parallel computations with different ODEs in-between time stamps like in Equation (8.1) since ODE solvers are hardly batch-parallelizable. We have circumvented this issue by replacing these ODEs with simple residual dynamics between time stamps, similarly to the Euler discretization:

$$\boldsymbol{y}_t = \boldsymbol{y}_{t_i} + (t - t_i) f_\theta(\boldsymbol{y}_{t_i}, \boldsymbol{z}_{i+1}). \tag{8.3}$$

This maintains the continuous-time view allowing the existence of the proposed adaptive stochastic model, while enabling us to perform batch-parallel computations thanks to the residual operation of Equation (8.3) which can be synchronized with the index i. This residual version also provides the opportunity for the temporal model to learn the optimal Euler discretization of Equation (8.1) by seeing the step size τ_i as the Euler step size Δt. Since τ_i may be greater than 1, unlike in SRVP, it paves the way for a lighter temporal model, which would save computations whenever no event in the input sequence requires a change of dynamics. With the Moving MNIST dataset, for instance, such a model would ideally synchronize the time stamps t_i with the digits' bounces and jump to the next bounce instead of allocating several superfluous residual operations in-between bounces.

However, learning this adaptive model has proved to be challenging. Indeed, variational learning is poorly adapted to this adaptive sampling strategy because the number of variables to infer vary depending on the training sequences and model parameters. Let us iteratively derive a possible ELBO for this model; we ignore the content variable $\boldsymbol{w}$ for the sake of clarity. First, we can infer the initial condition $\boldsymbol{y}_1$ like in SRVP:

$$\log p(\boldsymbol{x}) \geq \mathbb{E}_{\boldsymbol{y}_1 \sim q(\boldsymbol{y}_1 \mid \boldsymbol{x})}\big[\log p(\boldsymbol{x} \mid \boldsymbol{y}_1)\big] - D_{\mathrm{KL}}\big(q(\boldsymbol{y}_1 \mid \boldsymbol{x}) \,\big\|\, p(\boldsymbol{y}_1)\big). \tag{8.4}$$

We can then infer the first random variable $\boldsymbol{z}_2$:

$$\begin{aligned}\log p(\boldsymbol{x} \mid \boldsymbol{y}_1) \geq \mathbb{E}_{\boldsymbol{z}_2 \sim q(\boldsymbol{z}_2 \mid \boldsymbol{x}, \boldsymbol{y}_1)}\Big[\log p\big(\boldsymbol{x} \;\big|\; \boldsymbol{y}_1, \boldsymbol{y}_{t_2} = \boldsymbol{y}_1 + d_\vartheta(\boldsymbol{y}_1, \boldsymbol{z}_2) f_\theta(\boldsymbol{y}_1, \boldsymbol{z}_2)\big)\Big] \\ - D_{\mathrm{KL}}\big(q(\boldsymbol{z}_2 \mid \boldsymbol{x}, \boldsymbol{y}_1) \,\big\|\, p(\boldsymbol{z}_2 \mid \boldsymbol{y}_1)\big).\end{aligned} \tag{8.5}$$

A similar ELBO can then be derived for any $\log p\big(\boldsymbol{x} \mid \boldsymbol{y}_{t_k}, k \in [\![1, i]\!]\big)$, with $t_1 = 1$. We can iterate over these ELBOs until $\log p\big(\boldsymbol{x} \mid \boldsymbol{y}_{t_k}, k \in [\![1, i]\!]\big)$ can be analytically computed, i.e. when $t_i \geq T$, where T is the length of the sequence. This gives the following ELBO:

$$\begin{aligned}\log p(\boldsymbol{x}) \geq \mathbb{E}_{\substack{y_1 = y_{t_1}, (\boldsymbol{z}_i, \boldsymbol{y}_{t_i})_{i>1} \sim q_{Z,Y} \\ \text{such that } t_{i-1} < T \leq t_i}} \Bigg[& \log p\Big(\boldsymbol{x} \;\Big|\; (\boldsymbol{y}_{t_k})_{k \in [\![1,i]\!]}\Big) \\ & - \sum_{k=2}^{i} D_{\mathrm{KL}}\Big(q\big(\boldsymbol{z}_k \mid \boldsymbol{x}, \boldsymbol{y}_{t_{k-1}}\big) \;\Big\|\; p\big(\boldsymbol{z}_k \mid \boldsymbol{y}_{t_{k-1}}\big)\Big) \Bigg] \\ & - D_{\mathrm{KL}}\big(q(\boldsymbol{y}_1 \mid \boldsymbol{x}) \;\big\|\; p(\boldsymbol{y}_1)\big).\end{aligned} \tag{8.6}$$

Note that, unlike in SRVP, the inference of $\boldsymbol{z}_i$ necessitates both the data $\boldsymbol{x}$ and the current state $\boldsymbol{y}_{t_{i-1}}$, because the temporal model may not know the state of the system $\boldsymbol{y}_{t_i}$ corresponding to $\boldsymbol{z}_i$ when, in an adaptive setting, the time stamp of $\boldsymbol{z}_i$ varies throughout learning. The ELBO of Equation (8.6) is atypical as it involves a variable number of KLDs depending on the data point and the model parameters. We believe that this unusual variational inference has raised a number of optimization problems that could explain the difficulty that we have encountered to make this model work as intended, as described in the rest of this subsection.

We have indeed implemented the described model in various forms but have failed to prove its relevance in practice. The first encountered issue with the above model is its instability. Indeed, variational inference benefits during the first stage of training from increasing the number of variables to infer $\boldsymbol{z}$ for a better reconstruction of $\boldsymbol{x}$, which the model is able to do by decreasing the delays τ_i, leading to prohibitive time and memory costs. This has incited us to complement this adaptive model with a regularization term to minimize that penalizes overly small τ_i, of the form:

$$\lambda_1 \sum_i \frac{1}{\tau_i^{\lambda_2}}, \tag{8.7}$$

where λ_1 and λ_2 are positive hyperparameters. Besides solving the aforementioned computational cost explosion issue, the weighting of this penalty offers the possibility to balance the performance of a model with a soft constraint on the budget of the number of residual steps.

However, the experimental results of the model with this regularization have not been satisfying. Whatever the tested hyperparameter values, allocated budget and resulting number of residual steps used by the adaptive model, its performance on Moving MNIST has remained under SRVP. Moreover, we have not noticed any possibility of saving computations by learning large step sizes $\tau_i > 1$: the model tends to allocate many small residual steps at regular moments in the sequence, especially at its beginning, even with a large λ_1. This low prediction performance occurs even though the adaptive model does perform well when it comes to sequence reconstruction during training. This highlights the particularity of sequential latent variable models which are tested in our context

for prediction but trained with a different objective due to their variational training. We believe that introducing adaptive stochasticity in the investigated model widens the gap between training and testing even further, thereby penalizing its downstream prediction performance.

Nonetheless, we have found the results encouraging because qualitative analysis indicates that, despite its observed flaws, the adaptive model does detect bounces in the Moving MNIST data and is able, although not consistently, to increase the delays τ_i in-between bounces. Therefore, we would be interested in further investigating this research direction, especially as it has received some attention since then, with a few other concurrent adaptive ODE-based works in a deterministic and more simplistic setting than video prediction (R. T. Q. Chen, B. Amos, and Nickel, 2021; Sam Greydanus, S. Lee, and Fern, 2021).

8.1.2. GAN Improvements via the GAN(TK)^2 Framework

Along with the development of our GAN(TK)^2 framework in Chapter 6, we have attempted to leverage the new understanding provided by our framework in order to directly improve GANs. We have investigated with preliminary experiments two possibilities of improvement, exposed in the following.

8.1.2.1. New Discriminator Architectures

We have studied in Section 6.6 the NTK of some standard discriminator architectures. Subsequently, we have also tried to derive new, more performant, discriminator architectures by studying the properties of their NTKs.

We have been more specifically interested in the activations used in the discriminators. Indeed, the NTK k of a ReLU network, given in Equation (D.133), resembles no standard kernel: it is not translation-invariant and $k(\cdot, \boldsymbol{y})$ does not reach its maximum in the vicinity of $\boldsymbol{y}$, but it can rather be unbounded when its input norm tends to infinity. We have shown in Section 6.6 that this unusual kernel endows its corresponding discriminator with interesting generative properties. Still, we have wondered whether complementing it with other activation functions in a multi-branch architecture could improve its ability to drive the generator.

We have noticed, for example from the Neural Tangents library (Novak et al., 2020), that using sine-like activations leads to an NTK resembling an RBF, which could be interesting to combine with a ReLU NTK. Preliminary experiments have shown that using sine-like activations in conjunction with ReLU does give promising results on our two-dimensional problems introduced in Section 6.6.2: it experimentally removes the need for the discriminator to have non-null bias terms for the generated distribution to converge towards the target. However, we have not found conclusive evidence that such improvement transfers in the finite-width regime to widespread state-of-the-art discriminator architectures for images like DCGAN (Radford, Metz, and Chintala, 2016) and BigGAN (Brock, Donahue, and Simonyan, 2019). We have not been able to confirm the reason for this absence of results, given that such architectures are significantly more costly to manipulate in the infinite-width regime.

Nonetheless, we hope that more involved theoretical and experimental analyses of the NTKs of such architectures could lead to the discovery of more performant architectures. Regarding the introduction of sine-like activations, we believe that the recent advances that they have made possible in other domains thanks to their interesting properties support our motivation to introduce them in a GAN setting; see for example the works of Tancik et al. (2020) who also leverage the NTK theory, and of Sitzmann et al. (2020).

8.1.2.2. New NTK-Based GAN Model

In Section 6.5.1, we discover that, in the infinite-width regime of the discriminator, the generator under the IPM loss is actually trained to minimize the MMD between the generated and target distribution with respect to the NTK of the discriminator's architecture. We have sought to determine how such a generator trained with the NTK MMD witness function as a discriminator instead of a trained neural network would perform in practice.

Such a GAN model raises the issue of the computational cost of kernels compared to neural network discriminators. Indeed, in this case, the discriminator must be computed from scratch for each new instantiation of generator parameters. Moreover, this computation relies on a Monte-Carlo estimation using mini-batches of the generated and target distribution because of the quadratic cost in the number of samples to compute this MMD witness function. Preliminary tests of this model on MNIST have shown that this high-variance estimation hurts the performance of the generator. In particular, while increasing the size of the mini-batches improves its performance, we could not match the results of a standard DCGAN (Radford, Metz, and Chintala, 2016) within a reasonable computational budget.

Nevertheless, we believe that such an NTK MMD model is promising. We have found that it leads to a significantly more stable generator training since it removes the need of solving via gradient descent the inner optimization problem of the discriminator, i.e. it removes the adversarial component of GAN optimization. This finding is consistent with how NTKs have been used in a meta-learning setting (Y. Zhou et al., 2021). Should its computational issues be mitigated, this NTK GAN model could become a performant discriminator-free generative model. To this end, one may consider exploring various preexisting strategies to more efficiently compute the MMD witness functions (Gretton, Borgwardt, M. J. Rasch, et al., 2012; Muandet et al., 2017).

8.2. Future Directions

8.2.1. Temporal Data and Text

We take inspiration in Chapter 3 from a standard NLP technique to propose a representation learning method for time series. This inspiration from NLP has been beneficial but rests on word2vec (Mikolov, Sutskever, et al., 2013), which has become obsolete since then. In particular, the versatility of the transformer architecture mentioned in Section 2.1.3.2 constitutes a solid motivation to adapt them for applications on temporal data, especially given recent advances illustrating their ability to simultaneously work

on both modalities (Pashevich, Schmid, and C. Sun, 2021). More generally, advances in NLP might be a fruitful source of inspiration for machine learning research on temporal data due to the sequential nature of language. Conversely, we would be interested in applying some of the ideas in predictive and generative modeling developed in this document to textual data in order to evaluate the utility of differential equations at modeling them.

8.2.2. Spatiotemporal Prediction

We consider in this subsection axes of improvement for the spatiotemporal prediction models of Chapters 4 and 5.

8.2.2.1. Merging the Video and PDE-Based Models

Both chapters explore two different aspects of learning on spatiotemporal data and share similarities in model structures. Given the disentanglement improvements made in Chapter 5 compared to the more complex model of Chapter 4, we think that an interesting future work would be to merge these models by proposing a sequential latent variable model with a PDE-inspired spatiotemporal disentanglement system.

8.2.2.2. Scaling Models

Despite its state-of-the-art performance when released, the SRVP model of Chapter 4 has now been surpassed by other prediction methods. However, the motivation behind SRVP is not exclusively to achieve state-of-the-art results but also to present a new learning principle for video prediction with more advantages than prior methods.

Another way to study the advantages of SRVP would be to scale it with higher-capacity networks and architectures so as to determine whether its residual and state-space principles also benefit computationally costly methods. A positive result would be in accordance with the study of Castrejon, Ballas, and Courville (2019) and Villegas, Pathak, et al. (2019) who show that the older SVG model from Denton and Fergus (2018) does present a significantly increased performance when properly scaled.

8.2.2.3. Relaxing the Constancy of the Content Variable

We introduce in both chapters a content variable $\boldsymbol{w}$, representing the static content in a sequence. However, this assumption of a static component in the dynamics of a sequence usually holds for short or specific sequences only, because most partially observed spatiotemporal phenomena may involve the addition of new content in the course of a sequence. For example in videos, a new subject may appear or entirely new visuals might be introduced when a video is edited, which invalidates the purpose of this content variable. Indeed, it is likely that our models would not perform satisfyingly on more complex video datasets such as Kinetics-600 (Carreira et al., 2018) that is already challenging for high-capacity models (Weissenborn, Täckström, and Uszkoreit, 2020), partly because of this content variable.

A possible improvement would be to relax the constancy assumption of this variable. An idea would consist in making it evolve through time, although at a lesser rate than the state $\boldsymbol{y}$ via regularization strategies. Another possibility to explore would be to allow it to abruptly change at learned specific time steps, corresponding to abrupt changes of content in a video for instance, but forbid it to evolve the rest of the time.

8.2.3. NTKs for the Analysis of Generative Models

Finally, we consider here two possible follow-ups for our work on GAN analysis with NTKs in Chapter 6.

8.2.3.1. Analysis of GANs's Generators

Chapter 6 focuses on studying the discriminator because, as argued in Section 6.3.1, it is the basic building block of GANs. Furthermore, we show in the chapter that understanding the role of the generator in GAN optimization remains an open problem. Improving our understanding of the dynamics of the generated distribution through a further study of Equation (6.12), even for specific GAN losses, would be a significant step towards elucidating GANs' behavior.

8.2.3.2. Analysis of Other Models

A more exploratory direction could consist in studying other generative models through the prism of NTKs. Some might share similarities with GANs, which could enable us to transfer some results from Chapter 6. For example, score-based generative modeling involves in recent developments generated samples following the gradient of a neural network trained on a specific task (Y. Song et al., 2021). NTKs might prove useful to better understand the influence of neural networks architectures in such models.

Appendix

Appendix A.

Supplementary Material of Chapter 3

In this appendix chapter of Chapter 3, we provide our detailed training procedure for classification tasks, choices of hyperparameters, as well as the full experimental results of our method, compared to those of concurrent methods. Appendix A.1 further specifies the training process for classification tasks and details the choices of hyperparameters in all presented experiments. Appendix A.2 reports accuracy scores of all variants of our method on the whole UCR archive (Dau et al., 2018) as well as comparisons with concurrent methods, when available. Appendix A.3 provides accuracy scores for our method on the whole UEA archive (Bagnall, Dau, et al., 2018).

A.1. Training Details

A.1.1. Input Preprocessing

We preprocess datasets of the UCR archive that are not already normalized in their original version, as well as the IHEPC dataset, so that the set of time series values for each dataset has zero mean and unit variance. For each UEA dataset, each dimension of the time series is preprocessed independently of the other dimensions by normalizing in the same way its mean and variance.

A.1.2. SVM Training

In order to train an SVM on the computed representations of the elements of the train set, we perform a hyperparameter optimization for the penalty C of the error term of the SVM by cross-validating it over the representations of the train set, thus only using the train labels. Note that if the train set or the number of training samples per class are too small, we choose a penalty $C = \infty$ for the SVM (which corresponds to no regularization).

A.1.3. Hyperparameters

We train our models with the following parameters for time series classification. We emphasize that no hyperparameter optimization was performed on the encoder hyperparameters.

- Optimizer: Adam (Kingma and Ba, 2015) with learning rate $\alpha = 0.001$ and decay rates $\beta = (0.9, 0.999)$.
- SVM: penalty $C \in \left\{10^i \mid i \in [\![-4, 4]\!]\right\} \cup \{\infty\}$.
- Encoder training:
  - number of negative samples: $K \in \{1, 2, 5, 10\}$ for univariate time series, $K \in \{5, 10, 20\}$ for multivariate ones;
  - batch size: 10;
  - number of optimizations steps: 2000 for $K \geq 10$ (i.e., 20 epochs for a dataset of size 1000), 1500 otherwise.
- Architecture:
  - number of channels in the intermediary layers of the causal network: 40;
  - number of layers (depth of the causal network): 10;
  - kernel size of all convolutions: 3;
  - negative slope of the leaky ReLU activation: 0.01;
  - number of output channels of the causal network (before max pooling): 320;
  - dimension of the representations: 160.

For the Individual Household Electric Power Consumption dataset, changes are the following:

- number of negative samples: $K = 10$;
- batch size: 1;
- number of optimization steps: 400;
- number of channels in the intermediary layers of the causal network: 30;
- number of output channels of the causal network (before max pooling): 160;
- dimension of the representations: 80.

A.2. Univariate Time Series

Full results corresponding to the first 85 UCR datasets for our method are presented in Table A.1, while comparisons with DTW, ST, BOSS, HIVE-COTE and EE are shown in Figures 3.4 and 3.5 and Table A.2,[1] and comparisons with ResNet,[2] TimeNet and RWS are shown in Table A.3. Table A.4 compiles the results of our method and of DTW[3] for the newest 43 UCR datasets (except DodgerLoopDay, DodgerLoopGame and DodgerLoopWeekend which contain missing values).

[1] Scores taken from `http://www.timeseriesclassification.com/singleTrainTest.csv`.
[2] Scores taken from `https://github.com/hfawaz/dl-4-tsc/blob/master/results/results-uea.csv` (first iteration).
[3] Scores taken from `https://www.cs.ucr.edu/~eamonn/time_series_data_2018/`.

Table A.1.: Accuracy scores of variants of our method on the first 85 UCR datasets. "Combined (1-NN)" corresponds to learning a one-nearest-neighbor classifier, instead of an SVM, on the combined representations. Bold scores indicate the best performing method.

Dataset	Unsupervised						
	Ours						
	$K = 1$	$K = 2$	$K = 5$	$K = 10$	Combined	Combined (1-NN)	FordA
Adiac	0.734	0.711	0.703	0.675	0.716	0.645	**0.760**
ArrowHead	**0.869**	0.829	0.754	0.766	0.829	0.817	0.817
Beef	**0.733**	0.567	0.700	0.667	0.700	0.600	0.667
BeetleFly	**0.900**	0.800	**0.900**	0.800	**0.900**	0.800	0.800
BirdChicken	0.700	0.800	**0.900**	0.850	0.800	0.750	**0.900**
Car	0.750	0.767	0.633	0.833	0.817	0.800	**0.850**
CBF	0.982	0.991	0.990	0.983	**0.994**	0.978	0.988
ChlorineConcentration	0.719	0.747	0.739	0.749	**0.782**	0.588	0.688
CinCECGTorso	0.702	**0.747**	0.682	0.713	0.740	0.693	0.638
Coffee	0.964	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
Computers	**0.688**	0.644	0.676	0.664	0.628	0.604	0.648
CricketX	0.736	0.710	0.700	0.713	**0.777**	0.741	0.682
CricketY	0.682	0.664	0.695	0.728	**0.767**	0.664	0.667
CricketZ	0.721	0.710	0.726	0.708	**0.764**	0.723	0.656
DiatomSizeReduction	0.990	0.987	**0.993**	0.984	**0.993**	0.967	0.974
DistalPhalanxOutlineCorrect	0.761	0.746	**0.775**	**0.775**	0.768	0.757	0.764
DistalPhalanxOutlineAgeGroup	0.719	**0.748**	0.719	0.727	0.734	0.683	0.727
DistalPhalanxTW	**0.698**	0.676	0.662	0.676	0.676	0.669	0.669
Earthquakes	**0.748**	**0.748**	**0.748**	**0.748**	**0.748**	0.640	**0.748**
ECG200	0.870	0.900	0.860	**0.940**	0.900	0.850	0.830
ECG5000	0.939	0.939	0.937	0.933	0.936	0.925	**0.940**
ECGFiveDays	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.999	**1.000**

Table A.1.: Accuracy scores of variants of our method on the first 85 UCR datasets. "Combined (1-NN)" corresponds to learning a one-nearest-neighbor classifier, instead of an SVM, on the combined representations. Bold scores indicate the best performing method.

Dataset	Unsupervised						
	Ours						
	$K = 1$	$K = 2$	$K = 5$	$K = 10$	Combined	Combined (1-NN)	FordA
ElectricDevices	0.709	0.700	0.712	0.707	**0.732**	0.646	0.676
FaceAll	0.764	**0.810**	0.733	0.786	0.802	0.750	0.734
FaceFour	0.807	0.864	0.795	**0.920**	0.875	0.864	0.830
FacesUCR	0.885	0.871	0.886	0.884	**0.918**	0.860	0.835
FiftyWords	0.763	0.734	0.727	0.732	**0.780**	0.716	0.745
Fish	0.903	0.909	0.891	0.891	0.880	0.823	**0.960**
FordA	0.923	0.922	0.927	0.928	**0.935**	0.863	0.927
FordB	0.786	0.788	0.781	0.793	**0.810**	0.748	0.798
GunPoint	0.953	0.987	0.987	0.980	**0.993**	0.833	0.987
Ham	0.648	0.686	0.657	**0.724**	0.695	0.533	0.533
HandOutlines	**0.922**	0.919	0.908	**0.922**	**0.922**	0.832	0.919
Haptics	0.445	0.435	0.432	**0.490**	0.455	0.354	0.474
Herring	**0.609**	0.594	0.578	0.594	0.578	0.563	0.578
InlineSkate	0.425	0.429	0.427	0.371	**0.447**	0.400	0.444
InsectWingbeatSound	0.610	0.592	0.617	0.597	**0.623**	0.506	0.599
ItalyPowerDemand	0.940	0.927	0.928	**0.954**	0.925	0.942	0.929
LargeKitchenAppliances	0.797	0.827	0.843	0.789	**0.848**	0.757	0.765
Lightning2	0.869	0.836	0.852	0.869	**0.918**	0.885	0.787
Lightning7	0.795	**0.822**	**0.822**	0.795	0.795	0.795	0.740
Mallat	0.962	0.931	0.947	0.951	**0.964**	0.944	0.916
Meat	0.917	0.867	0.867	**0.950**	**0.950**	0.900	0.867
MedicalImages	0.738	0.768	0.770	0.750	**0.784**	0.693	0.725

Table A.1.: Accuracy scores of variants of our method on the first 85 UCR datasets. "Combined (1-NN)" corresponds to learning a one-nearest-neighbor classifier, instead of an SVM, on the combined representations. Bold scores indicate the best performing method.

Dataset	Unsupervised						
	Ours						
	$K = 1$	$K = 2$	$K = 5$	$K = 10$	Combined	Combined (1-NN)	FordA
MiddlePhalanxOutlineCorrect	0.749	0.818	0.777	**0.825**	0.814	0.722	0.787
MiddlePhalanxOutlineAgeGroup	0.617	**0.662**	0.656	0.656	0.656	0.506	0.623
MiddlePhalanxTW	0.604	**0.610**	**0.610**	0.591	**0.610**	0.513	0.584
MoteStrain	**0.875**	0.854	0.867	0.851	0.871	0.853	0.823
NonInvasiveFetalECGThorax1	0.912	0.911	0.904	0.878	0.910	0.798	**0.925**
NonInvasiveFetalECGThorax2	0.925	0.925	0.918	0.919	0.927	0.820	**0.930**
OliveOil	0.867	0.833	0.867	0.867	**0.900**	0.833	**0.900**
OSULeaf	0.719	0.694	0.793	0.760	**0.831**	0.636	0.736
PhalangesOutlinesCorrect	**0.807**	0.796	0.795	0.784	0.801	0.752	0.784
Phoneme	0.264	0.265	0.249	0.276	**0.289**	0.197	0.196
Plane	0.990	**1.000**	0.990	0.990	0.990	**1.000**	0.981
ProximalPhalanxOutlineCorrect	**0.869**	0.863	0.856	0.859	0.859	0.801	**0.869**
ProximalPhalanxOutlineAgeGroup	0.849	**0.859**	0.844	0.844	0.854	0.805	0.839
ProximalPhalanxTW	**0.824**	0.815	0.761	0.771	**0.824**	0.717	0.785
RefrigerationDevices	0.531	0.507	0.547	0.515	0.517	0.475	**0.555**
ScreenType	0.408	0.411	**0.427**	0.416	0.413	0.389	0.384
ShapeletSim	**0.894**	0.500	0.628	0.672	0.817	0.772	0.517
ShapesAll	0.847	0.840	0.857	0.848	**0.875**	0.823	0.837
SmallKitchenAppliances	0.680	0.667	0.715	0.677	0.715	0.619	**0.731**
SonyAIBORobotSurface1	**0.930**	0.890	0.850	0.902	0.897	0.825	0.840
SonyAIBORobotSurface2	0.885	0.933	0.928	0.889	**0.934**	0.885	0.832
StarLightCurves	0.960	0.966	0.958	0.964	0.965	0.893	**0.968**

Table A.1.: Accuracy scores of variants of our method on the first 85 UCR datasets. "Combined (1-NN)" corresponds to learning a one-nearest-neighbor classifier, instead of an SVM, on the combined representations. Bold scores indicate the best performing method.

Dataset	Unsupervised						
	Ours						
	$K = 1$	$K = 2$	$K = 5$	$K = 10$	Combined	Combined (1-NN)	FordA
Strawberry	0.951	0.946	**0.954**	**0.954**	0.946	0.903	0.946
SwedishLeaf	0.907	0.925	0.925	0.914	**0.931**	0.891	0.925
Symbols	0.937	0.931	**0.965**	0.963	**0.965**	0.933	0.945
SyntheticControl	0.980	0.983	**0.987**	**0.987**	0.983	0.977	0.977
ToeSegmentation1	0.868	**0.961**	0.930	0.939	0.952	0.851	0.899
ToeSegmentation2	0.869	0.892	0.838	**0.900**	0.885	**0.900**	**0.900**
Trace	**1.000**	**1.000**	**1.000**	0.990	**1.000**	**1.000**	**1.000**
TwoLeadECG	0.996	0.991	0.996	**0.999**	0.997	0.988	0.993
TwoPatterns	0.998	**1.000**	**1.000**	0.999	**1.000**	0.998	0.992
UWaveGestureLibraryX	0.795	0.791	0.806	0.785	**0.811**	0.762	0.784
UWaveGestureLibraryY	0.716	0.717	0.702	0.710	**0.735**	0.666	0.697
UWaveGestureLibraryZ	0.738	0.735	0.741	0.757	**0.759**	0.679	0.729
UWaveGestureLibraryAll	0.893	0.887	0.903	0.896	**0.941**	0.838	0.865
Wafer	0.991	**0.995**	0.993	0.992	0.993	0.987	**0.995**
Wine	0.704	0.815	0.852	0.815	**0.870**	0.500	0.685
WordSynonyms	0.630	0.646	0.676	0.691	**0.704**	0.633	0.641
Worms	0.662	**0.740**	0.688	0.727	0.714	0.597	0.688
WormsTwoClass	0.753	0.766	0.740	0.792	**0.818**	0.805	0.753
Yoga	0.824	0.854	0.831	0.837	**0.878**	0.837	0.828

Table A.2.: Accuracy scores of the combined version of our method compared with those of DTW (unsupervised), ST and BOSS (supervised) and HIVE-COTE and EE (supervised ensemble methods), on the first 85 UCR datasets (results on the full archive were not available for comparisons). Bold scores indicate the best performing method.

Dataset	Unsupervised		Supervised			
	Ours	DTW	ST	BOSS	Ensemble	
	Combined				HIVE-COTE	EE
Adiac	0.716	0.604	0.783	0.765	**0.811**	0.665
ArrowHead	0.829	0.703	0.737	0.834	**0.863**	0.811
Beef	0.700	0.633	0.900	0.800	**0.933**	0.633
BeetleFly	0.900	0.700	0.900	0.900	**0.950**	0.750
BirdChicken	0.800	0.750	0.800	**0.950**	0.850	0.800
Car	0.817	0.733	**0.917**	0.833	0.867	0.833
CBF	0.994	0.997	0.974	0.998	**0.999**	0.998
ChlorineConcentration	**0.782**	0.648	0.700	0.661	0.712	0.656
CinCECGTorso	0.740	0.651	0.954	0.887	**0.996**	0.942
Coffee	**1.000**	**1.000**	0.964	**1.000**	**1.000**	**1.000**
Computers	0.628	0.700	0.736	0.756	**0.760**	0.708
CricketX	0.777	0.754	0.772	0.736	**0.823**	0.813
CricketY	0.767	0.744	0.779	0.754	**0.849**	0.805
CricketZ	0.764	0.754	0.787	0.746	**0.831**	0.782
DiatomSizeReduction	**0.993**	0.967	0.925	0.931	0.941	0.944
DistalPhalanxOutlineCorrect	0.768	0.717	**0.775**	0.728	0.772	0.728
DistalPhalanxOutlineAgeGroup	0.734	**0.770**	**0.770**	0.748	0.763	0.691
DistalPhalanxTW	0.676	0.590	0.662	0.676	**0.683**	0.647
Earthquakes	**0.748**	0.719	0.741	**0.748**	**0.748**	0.741
ECG200	**0.900**	0.770	0.830	0.870	0.850	0.880
ECG5000	0.936	0.924	0.944	0.941	**0.946**	0.939
ECGFiveDays	**1.000**	0.768	0.984	**1.000**	**1.000**	0.820

Table A.2.: Accuracy scores of the combined version of our method compared with those of DTW (unsupervised), ST and BOSS (supervised) and HIVE-COTE and EE (supervised ensemble methods), on the first 85 UCR datasets (results on the full archive were not available for comparisons). Bold scores indicate the best performing method.

Dataset	Unsupervised		Supervised			
	Ours	DTW	ST	BOSS	Ensemble	
	Combined				HIVE-COTE	EE
ElectricDevices	0.732	0.602	0.747	**0.799**	0.770	0.663
FaceAll	0.802	0.808	0.779	0.782	0.803	**0.849**
FaceFour	0.875	0.830	0.852	**1.000**	0.955	0.909
FacesUCR	0.918	0.905	0.906	0.957	**0.963**	0.945
FiftyWords	0.780	0.690	0.705	0.705	0.809	**0.820**
Fish	0.880	0.823	**0.989**	**0.989**	**0.989**	0.966
FordA	0.935	0.555	0.971	0.930	**0.964**	0.738
FordB	0.810	0.620	0.807	0.711	**0.823**	0.662
GunPoint	0.993	0.907	**1.000**	**1.000**	**1.000**	0.993
Ham	**0.695**	0.467	0.686	0.667	0.667	0.571
HandOutlines	0.922	0.881	**0.932**	0.903	**0.932**	0.889
Haptics	0.455	0.377	**0.523**	0.461	0.519	0.393
Herring	0.578	0.531	0.672	0.547	**0.688**	0.578
InlineSkate	0.447	0.384	0.373	**0.516**	0.500	0.460
InsectWingbeatSound	0.623	0.355	0.627	0.523	**0.655**	0.595
ItalyPowerDemand	0.925	0.950	0.948	0.909	**0.963**	0.962
LargeKitchenAppliances	0.848	0.795	0.859	0.765	**0.864**	0.811
Lightning2	**0.918**	0.869	0.738	0.836	0.820	0.885
Lightning7	**0.795**	0.726	0.726	0.685	0.740	0.767
Mallat	**0.964**	0.934	**0.964**	0.938	0.962	0.940
Meat	**0.950**	0.933	0.850	0.900	0.933	0.933
MedicalImages	**0.784**	0.737	0.670	0.718	0.778	0.742

Table A.2.: Accuracy scores of the combined version of our method compared with those of DTW (unsupervised), ST and BOSS (supervised) and HIVE-COTE and EE (supervised ensemble methods), on the first 85 UCR datasets (results on the full archive were not available for comparisons). Bold scores indicate the best performing method.

Dataset	Unsupervised		Supervised			
	Ours	DTW	ST	BOSS	Ensemble	
	Combined				HIVE-COTE	EE
MiddlePhalanxOutlineCorrect	0.814	0.698	0.794	0.780	**0.832**	0.784
MiddlePhalanxOutlineAgeGroup	**0.656**	0.500	0.643	0.545	0.597	0.558
MiddlePhalanxTW	**0.610**	0.506	0.519	0.545	0.571	0.513
MoteStrain	0.871	0.835	0.897	0.879	**0.933**	0.883
NonInvasiveFetalECGThorax1	0.910	0.790	**0.950**	0.838	0.930	0.846
NonInvasiveFetalECGThorax2	0.927	0.865	**0.951**	0.901	0.945	0.913
OliveOil	**0.900**	0.833	**0.900**	0.867	**0.900**	0.867
OSULeaf	0.831	0.591	0.967	0.955	**0.979**	0.806
PhalangesOutlinesCorrect	0.801	0.728	0.763	0.772	**0.807**	0.773
Phoneme	0.289	0.228	0.321	0.265	**0.382**	0.305
Plane	0.990	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
ProximalPhalanxOutlineCorrect	0.859	0.784	**0.883**	0.849	0.880	0.808
ProximalPhalanxOutlineAgeGroup	0.854	0.805	0.844	0.834	**0.859**	0.805
ProximalPhalanxTW	**0.824**	0.761	0.805	0.800	0.815	0.766
RefrigerationDevices	0.517	0.464	**0.581**	0.499	0.557	0.437
ScreenType	0.413	0.397	0.520	0.464	**0.589**	0.445
ShapeletSim	0.817	0.650	0.956	**1.000**	**1.000**	0.817
ShapesAll	0.875	0.768	0.842	**0.908**	0.905	0.867
SmallKitchenAppliances	0.715	0.643	0.792	0.725	**0.853**	0.696
SonyAIBORobotSurface1	**0.897**	0.725	0.844	0.632	0.765	0.704
SonyAIBORobotSurface2	**0.934**	0.831	**0.934**	0.859	0.928	0.878
StarLightCurves	0.965	0.907	0.979	0.978	**0.982**	0.926

Table A.2.: Accuracy scores of the combined version of our method compared with those of DTW (unsupervised), ST and BOSS (supervised) and HIVE-COTE and EE (supervised ensemble methods), on the first 85 UCR datasets (results on the full archive were not available for comparisons). Bold scores indicate the best performing method.

Dataset	Unsupervised		Supervised			
	Ours	DTW	ST	BOSS	Ensemble	
	Combined				HIVE-COTE	EE
Strawberry	0.946	0.941	0.962	**0.976**	0.970	0.946
SwedishLeaf	0.931	0.792	0.928	0.922	**0.954**	0.915
Symbols	0.965	0.950	0.882	0.967	**0.974**	0.960
SyntheticControl	0.983	0.993	0.983	0.967	**0.997**	0.990
ToeSegmentation1	0.952	0.772	0.965	0.939	**0.982**	0.829
ToeSegmentation2	0.885	0.838	0.908	**0.962**	0.954	0.892
Trace	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.990
TwoLeadECG	**0.997**	0.905	**0.997**	0.981	0.996	0.971
TwoPatterns	**1.000**	**1.000**	0.955	0.993	**1.000**	**1.000**
UWaveGestureLibraryX	0.811	0.728	0.803	0.762	**0.840**	0.805
UWaveGestureLibraryY	0.735	0.634	0.730	0.685	**0.765**	0.726
UWaveGestureLibraryZ	0.759	0.658	0.748	0.695	**0.783**	0.724
UWaveGestureLibraryAll	0.941	0.892	0.942	0.939	**0.968**	**0.968**
Wafer	0.993	0.980	**1.000**	0.995	0.999	0.997
Wine	**0.870**	0.574	0.796	0.741	0.778	0.574
WordSynonyms	0.704	0.649	0.571	0.638	0.738	**0.779**
Worms	0.714	0.584	**0.740**	0.558	0.558	0.662
WormsTwoClass	0.818	0.623	**0.831**	**0.831**	0.779	0.688
Yoga	0.878	0.837	0.818	**0.918**	**0.918**	0.879

Table A.3.: Accuracy scores of the combined version of our method compared with those of ResNet (supervised), TimeNet and RWS (unsupervised), when available. Bold scores indicate the best performing method. "✕"s indicate that a score was reported in the original paper, but was either obtained using an encoder transfer from an other dataset or on a reversed train / test split of the dataset, thus not comparable to our results in these cases.

Dataset	Unsupervised	Supervised	Unsupervised	
	Ours	ResNet	TimeNet	RWS
	Combined			
Adiac	0.716	**0.831**	0.565	—
ArrowHead	0.829	**0.840**	—	—
Beef	0.700	**0.767**	—	0.733
BeetleFly	**0.900**	0.850	—	—
BirdChicken	0.800	**0.950**	—	—
Car	0.817	**0.917**	—	—
CBF	**0.994**	0.989	—	—
ChlorineConcentration	0.782	**0.835**	0.723	0.572
CinCECGTorso	0.740	**0.838**	—	—
Coffee	**1.000**	**1.000**	—	—
Computers	0.628	**0.816**	—	—
CricketX	0.777	**0.790**	0.659	—
CricketY	0.767	**0.805**	✕	—
CricketZ	0.764	**0.831**	✕	—
DiatomSizeReduction	**0.993**	0.301	—	—
DistalPhalanxOutlineCorrect	**0.768**	✕	✕	—
DistalPhalanxOutlineAgeGroup	**0.734**	✕	✕	—
DistalPhalanxTW	**0.676**	✕	✕	✕
Earthquakes	**0.748**	✕	—	—
ECG200	**0.900**	0.870	—	—
ECG5000	**0.936**	0.935	0.934	0.933

Table A.3.: Accuracy scores of the combined version of our method compared with those of ResNet (supervised), TimeNet and RWS (unsupervised), when available. Bold scores indicate the best performing method. "✕"s indicate that a score was reported in the original paper, but was either obtained using an encoder transfer from an other dataset or on a reversed train / test split of the dataset, thus not comparable to our results in these cases.

Dataset	Unsupervised	Supervised	Unsupervised	
	Ours	ResNet	TimeNet	RWS
	Combined			
ECGFiveDays	**1.000**	0.990	✕	—
ElectricDevices	0.732	**0.735**	0.665	—
FaceAll	0.802	**0.855**	—	—
FaceFour	0.875	**0.955**	—	—
FacesUCR	0.918	**0.955**	—	—
FiftyWords	**0.780**	0.732	—	—
Fish	0.880	**0.977**	—	—
FordA	**0.935**	✕	✕	—
FordB	**0.810**	✕	✕	✕
GunPoint	**0.993**	**0.993**	—	—
Ham	0.695	**0.800**	—	—
HandOutlines	**0.922**	✕	—	✕
Haptics	0.455	**0.516**	—	—
Herring	0.578	**0.641**	—	—
InlineSkate	**0.447**	0.378	—	—
InsectWingbeatSound	**0.623**	0.506	—	0.619
ItalyPowerDemand	0.925	0.959	—	**0.969**
LargeKitchenAppliances	0.848	**0.904**	—	0.792
Lightning2	**0.918**	0.770	—	—
Lightning7	0.795	**0.863**	—	—
Mallat	0.964	**0.966**	—	0.937

Table A.3.: Accuracy scores of the combined version of our method compared with those of ResNet (supervised), TimeNet and RWS (unsupervised), when available. Bold scores indicate the best performing method. "✕"s indicate that a score was reported in the original paper, but was either obtained using an encoder transfer from an other dataset or on a reversed train / test split of the dataset, thus not comparable to our results in these cases.

Dataset	Unsupervised	Supervised	Unsupervised	
	Ours	ResNet	TimeNet	RWS
	Combined			
Meat	0.950	**0.983**	—	—
MedicalImages	**0.784**	0.762	0.753	—
MiddlePhalanxOutlineCorrect	**0.814**	✕	✕	✕
MiddlePhalanxOutlineAgeGroup	**0.656**	✕	✕	—
MiddlePhalanxTW	**0.610**	✕	✕	—
MoteStrain	0.871	**0.924**	—	—
NonInvasiveFetalECGThorax1	0.910	**0.946**	—	0.907
NonInvasiveFetalECGThorax2	0.927	**0.944**	—	—
OliveOil	**0.900**	0.867	—	—
OSULeaf	0.831	**0.979**	—	—
PhalangesOutlinesCorrect	0.801	**0.857**	0.772	—
Phoneme	0.289	**0.333**	—	—
Plane	0.990	**1.000**	—	—
ProximalPhalanxOutlineCorrect	0.859	**0.914**	✕	0.711
ProximalPhalanxOutlineAgeGroup	**0.854**	0.839	✕	✕
ProximalPhalanxTW	**0.824**	✕	✕	—
RefrigerationDevices	**0.517**	**0.517**	—	—
ScreenType	0.413	**0.632**	—	—
ShapeletSim	0.817	**1.000**	—	—
ShapesAll	0.875	**0.917**	—	—
SmallKitchenAppliances	0.715	**0.789**	—	—

Table A.3.: Accuracy scores of the combined version of our method compared with those of ResNet (supervised), TimeNet and RWS (unsupervised), when available. Bold scores indicate the best performing method. "✕"s indicate that a score was reported in the original paper, but was either obtained using an encoder transfer from an other dataset or on a reversed train / test split of the dataset, thus not comparable to our results in these cases.

Dataset	Unsupervised	Supervised	Unsupervised	
	Ours	ResNet	TimeNet	RWS
	Combined			
SonyAIBORobotSurface1	0.897	**0.968**	—	—
SonyAIBORobotSurface2	0.934	**0.986**	—	—
StarLightCurves	0.965	**0.972**	—	—
Strawberry	**0.946**	✕	✕	—
SwedishLeaf	0.931	**0.955**	0.901	—
Symbols	**0.965**	0.927	—	—
SyntheticControl	0.983	**1.000**	0.983	—
ToeSegmentation1	0.952	**0.969**	—	—
ToeSegmentation2	0.885	**0.915**	—	—
Trace	**1.000**	**1.000**	—	—
TwoLeadECG	0.997	**1.000**	—	—
TwoPatterns	**1.000**	**1.000**	0.999	0.999
UWaveGestureLibraryX	**0.811**	0.780	—	—
UWaveGestureLibraryY	**0.735**	0.675	—	—
UWaveGestureLibraryZ	**0.759**	0.750	—	—
UWaveGestureLibraryAll	**0.941**	0.862	—	—
Wafer	0.993	**0.998**	0.994	0.993
Wine	**0.870**	0.611	—	—
WordSynonyms	**0.704**	0.625	—	—
Worms	**0.714**	✕	—	—
WormsTwoClass	**0.818**	✕	—	—

Table A.3.: Accuracy scores of the combined version of our method compared with those of ResNet (supervised), TimeNet and RWS (unsupervised), when available. Bold scores indicate the best performing method. "✗"s indicate that a score was reported in the original paper, but was either obtained using an encoder transfer from an other dataset or on a reversed train / test split of the dataset, thus not comparable to our results in these cases.

	Unsupervised	Supervised	Unsupervised	
Dataset	Ours	ResNet	TimeNet	RWS
	Combined			
Yoga	**0.878**	0.857	0.866	—

Table A.4.: Accuracy scores of variants of our method and of DTW on the remaining 43 UCR datasets, except DodgerLoopDay, DodgerLoopGame and DodgerLoopWeekend which contain missing values. Bold scores indicate the best performing method.

Dataset	Unsupervised							
	Ours							DTW
	$K = 1$	$K = 2$	$K = 5$	$K = 10$	Combined	Combined (1-NN)	FordA	
ACSF1	**0.910**	0.870	0.860	0.900	0.810	0.850	0.730	0.640
AllGestureWiimoteX	0.721	0.746	0.747	0.763	**0.779**	0.736	0.693	0.716
AllGestureWiimoteY	0.741	0.744	0.759	0.726	**0.793**	0.756	0.713	0.729
AllGestureWiimoteZ	0.687	0.697	0.691	0.723	**0.763**	0.716	0.710	0.643
BME	**0.993**	**0.993**	0.987	**0.993**	**0.993**	0.947	0.960	0.900
Chinatown	0.951	0.951	0.942	0.951	**0.962**	0.936	**0.962**	0.957
Crop	0.728	0.726	0.728	0.722	**0.746**	0.695	0.727	0.665
EOGHorizontalSignal	0.552	0.566	0.536	**0.605**	0.588	0.522	0.470	0.503
EOGVerticalSignal	0.398	0.414	0.431	0.434	**0.489**	0.431	0.439	0.448
EthanolLevel	0.418	0.340	0.316	0.382	0.392	0.274	**0.558**	0.276
FreezerRegularTrain	0.986	0.988	0.979	0.956	0.955	0.963	**0.992**	0.899
FreezerSmallTrain	0.967	0.956	0.906	**0.933**	0.928	0.872	0.862	0.753
Fungi	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.925	0.839
GestureMidAirD1	**0.638**	0.577	0.592	0.608	0.615	0.546	0.608	0.569
GestureMidAirD2	0.508	0.515	0.523	**0.546**	0.508	0.415	0.538	0.608
GestureMidAirD3	0.269	**0.331**	0.308	0.285	**0.331**	0.246	0.292	0.323
GesturePebbleZ1	0.826	0.843	0.913	0.919	**0.936**	0.814	0.547	0.791
GesturePebbleZ2	0.861	0.873	0.880	**0.899**	0.880	0.791	0.538	0.671
GunPointAgeSpan	0.984	0.984	**0.994**	**0.994**	0.987	0.991	0.987	0.918
GunPointMaleVersusFemale	**1.000**	**1.000**	**1.000**	0.997	**1.000**	0.994	**1.000**	0.997
GunPointOldVersusYoung	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.838
HouseTwenty	**0.950**	0.933	0.916	0.933	**0.950**	0.924	0.882	0.924

Table A.4.: Accuracy scores of variants of our method and of DTW on the remaining 43 UCR datasets, except DodgerLoopDay, DodgerLoopGame and DodgerLoopWeekend which contain missing values. Bold scores indicate the best performing method.

Dataset	Unsupervised							
	Ours							DTW
	$K = 1$	$K = 2$	$K = 5$	$K = 10$	Combined	Combined (1-NN)	FordA	
InsectEPGRegularTrain	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.872
InsectEPGSmallTrain	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.735
MelbournePedestrian	0.949	0.946	0.943	0.944	**0.951**	0.914	0.947	0.791
MixedShapesRegularTrain	0.916	0.906	0.904	0.905	**0.927**	0.898	0.898	0.842
MixedShapesSmallTrain	0.864	0.857	0.871	0.860	**0.877**	0.829	0.861	0.780
PickupGestureWiimoteZ	0.720	**0.800**	0.780	0.740	0.780	0.720	0.740	0.660
PigAirwayPressure	0.385	0.452	**0.510**	**0.510**	0.486	0.332	0.317	0.106
PigArtPressure	0.880	0.933	**0.942**	0.928	0.933	0.861	0.591	0.245
PigCVP	0.404	0.548	0.620	**0.788**	0.712	0.385	0.534	0.154
PLAID	0.533	0.549	0.574	0.555	0.559	0.696	0.493	**0.840**
PowerCons	**0.961**	0.939	0.900	0.900	0.928	0.894	0.933	0.878
Rock	0.620	0.620	0.580	0.580	**0.680**	0.500	0.540	0.600
SemgHandGenderCh2	0.845	0.852	0.873	0.890	**0.902**	0.863	0.840	0.802
SemgHandMovementCh2	0.711	0.649	0.700	**0.789**	0.784	0.709	0.516	0.584
SemgHandSubjectCh2	0.767	0.816	0.851	0.853	**0.876**	0.720	0.591	0.727
ShakeGestureWiimoteZ	0.920	0.920	**0.940**	0.920	**0.940**	0.860	0.900	0.860
SmoothSubspace	0.933	**0.960**	0.940	**0.960**	0.953	0.833	0.940	0.827
UMD	0.979	0.986	**0.993**	**0.993**	**0.993**	0.958	0.986	**0.993**

A.3. Multivariate Time Series

Full results corresponding to the UEA archive datasets for our method as well as the ones of DTW_D as reported by Bagnall, Dau, et al. (2018) are presented in Table A.5, for the unique train / test split provided in the archive.

Table A.5.: Accuracy scores of variants of our method on all UEA datasets, compared to DTW_D. Bold scores indicate the best performing method.

Dataset	Unsupervised				
	Ours				DTW_D
	$K = 5$	$K = 10$	$K = 20$	Combined	
ArticularyWordRecognition	0.967	0.973	0.943	**0.987**	**0.987**
AtrialFibrillation	**0.200**	0.067	0.133	0.133	**0.200**
BasicMotions	**1.000**	**1.000**	**1.000**	**1.000**	0.975
CharacterTrajectories	0.986	0.990	0.993	**0.994**	0.989
Cricket	0.958	0.972	0.972	0.986	**1.000**
DuckDuckGeese	0.600	**0.675**	0.650	**0.675**	0.600
EigenWorms	0.870	0.802	0.840	**0.878**	0.618
Epilepsy	**0.971**	**0.971**	**0.971**	0.957	0.964
Ering	**0.133**	**0.133**	**0.133**	**0.133**	**0.133**
EthanolConcentration	0.289	0.251	0.205	0.236	**0.323**
FaceDetection	0.522	0.525	0.513	0.528	**0.529**
FingerMovements	0.550	0.490	**0.580**	0.540	0.530
HandMovementDirection	0.311	0.297	**0.351**	0.270	0.231
Handwriting	0.447	0.464	0.451	**0.533**	0.286
Heartbeat	**0.756**	0.732	0.741	0.737	0.717
InsectWingbeat	0.159	0.158	0.156	**0.160**	—
JapaneseVowels	0.984	0.986	**0.989**	**0.989**	0.949
Libras	0.878	**0.883**	**0.883**	0.867	0.870
LSST	0.535	0.552	0.509	**0.558**	0.551
MotorImagery	0.530	0.540	**0.580**	0.540	0.500
NATOPS	0.933	0.917	0.917	**0.944**	0.883
PEMS-SF	0.636	0.671	0.676	**0.688**	0.711
PenDigits	**0.985**	0.979	0.981	0.983	0.977

Table A.5.: Accuracy scores of variants of our method on all UEA datasets, compared to DTW_D. Bold scores indicate the best performing method.

Dataset	Unsupervised				
	Ours				DTW_D
	$K = 5$	$K = 10$	$K = 20$	Combined	
Phoneme	0.216	0.214	0.222	**0.246**	0.151
RacketSports	0.776	0.836	0.855	**0.862**	0.803
SelfRegulationSCP1	0.795	0.826	0.843	**0.846**	0.775
SelfRegulationSCP2	0.550	0.539	0.539	**0.556**	0.539
SpokenArabicDigits	0.908	0.894	0.905	0.956	**0.963**
StandWalkJump	0.333	**0.400**	0.333	**0.400**	0.200
UWaveGestureLibrary	0.884	0.869	0.875	0.884	**0.903**

Appendix B.

Supplementary Material of Chapter 4

In this appendix chapter of Chapter 4, we provide supplementary details, explanations and discussions as well as additional experiments. Appendix B.1 derives in detail the model's ELBO of Equation (4.9). Appendix B.2 presents the skimmed datasets used in Section 4.4 and associated technical details. Appendix B.3 provides the full training details and hyperparameter choices required to reproduce our results. Appendix B.4 extends the discussion started in Section 4.4.3 on the ability of our model to produce predictions at arbitrary framerates. Appendix B.5 introduces an additional experiment involving our model without content variable for sequence modeling, instead of prediction in the main chapter. Finally, Appendix B.6 contains additional prediction samples for all datasets and compared methods in Section 4.4.

B.1. Evidence Lower Bound

We develop in this section the computations of the variational lower bound for the proposed model.

Using the original ELBO of Kingma and Welling (2014) and Equation (2.22) in Equation (B.1):

$$\begin{aligned}
&\log p(\boldsymbol{x}_{1:T} \mid \boldsymbol{w}) \\
&\quad \geq \mathbb{E}_{(\boldsymbol{z}_{2:T}, \boldsymbol{y}_{1:T}) \sim q_{Z,Y}} \log p(\boldsymbol{x}_{1:T} \mid \boldsymbol{z}_{2:T}, \boldsymbol{y}_{1:T}, \boldsymbol{w}) - D_{\mathrm{KL}}\big(q_{Z,Y} \,\big\|\, p(\boldsymbol{y}_{1:T}, \boldsymbol{z}_{2:T} \mid \boldsymbol{w})\big) && \text{(B.1)} \\
&\quad = \mathbb{E}_{(\boldsymbol{z}_{2:T}, \boldsymbol{y}_{1:T}) \sim q_{Z,Y}} \log p(\boldsymbol{x}_{1:T} \mid \boldsymbol{z}_{2:T}, \boldsymbol{y}_{1:T}, \boldsymbol{w}) \\
&\qquad\qquad - D_{\mathrm{KL}}\big(q(\boldsymbol{y}_1, \boldsymbol{z}_{2:T} \mid \boldsymbol{x}_{1:T}) \,\big\|\, p(\boldsymbol{y}_1, \boldsymbol{z}_{2:T})\big) && \text{(B.2)} \\
&\quad = \mathbb{E}_{(\boldsymbol{z}_{2:T}, \boldsymbol{y}_{1:T}) \sim q_{Z,Y}} \sum_{t=1}^{T} \log p(\boldsymbol{x}_t \mid \boldsymbol{y}_t, \boldsymbol{w}) - D_{\mathrm{KL}}\big(q(\boldsymbol{y}_1, \boldsymbol{z}_{2:T} \mid \boldsymbol{x}_{1:T}) \,\big\|\, p(\boldsymbol{y}_1, \boldsymbol{z}_{2:T})\big), && \text{(B.3)}
\end{aligned}$$

where:

- Equation (B.2) is given by the forward and inference models factorizing p and q in Equations (4.5), (4.6) and (4.8) and illustrated by, respectively, Figures 4.1(a) and 4.1(b):

  - the $\boldsymbol{z}$ variables and $\boldsymbol{y}_1$ are independent of $\boldsymbol{w}$ with respect to p and q;
  - the $\boldsymbol{y}_{2:T}$ variables are deterministic functions of $\boldsymbol{y}_1$ and $\boldsymbol{z}_{2:T}$ with respect to p and q;

- Equation (B.3) results from the factorization of $p(\boldsymbol{x}_{1:T} \mid \boldsymbol{y}_{1:T}, \boldsymbol{z}_{1:T}, \boldsymbol{w})$ in Equation (4.5).

From there, by using the integral formulation of D_{KL}:

$$
\begin{aligned}
&\log p(\boldsymbol{x}_{1:T} \mid \boldsymbol{w}) \\
&\quad \geq \mathbb{E}_{(\boldsymbol{z}_{2:T}, \boldsymbol{y}_{1:T}) \sim q_{Z,Y}} \sum_{t=1}^{T} \log p(\boldsymbol{x}_t \mid \boldsymbol{y}_t, \boldsymbol{w}) \\
&\qquad + \int \cdots \int_{\boldsymbol{y}_1, \boldsymbol{z}_{2:T}} q(\boldsymbol{y}_1, \boldsymbol{z}_{2:T} \mid \boldsymbol{x}_{1:T}) \log \frac{p(\boldsymbol{y}_1, \boldsymbol{z}_{2:T})}{q(\boldsymbol{y}_1, \boldsymbol{z}_{2:T} \mid \boldsymbol{x}_{1:T})} \, \mathrm{d}\boldsymbol{z}_{2:T} \, \mathrm{d}\boldsymbol{y}_1
\end{aligned} \tag{B.4}
$$

$$
\begin{aligned}
&= \mathbb{E}_{(\boldsymbol{z}_{2:T}, \boldsymbol{y}_{1:T}) \sim q_{Z,Y}} \sum_{t=1}^{T} \log p(\boldsymbol{x}_t \mid \boldsymbol{y}_t, \boldsymbol{w}) - D_{\mathrm{KL}}\big(q(\boldsymbol{y}_1 \mid \boldsymbol{x}_{1:T}) \,\big\|\, p(\boldsymbol{y}_1)\big) \\
&\qquad + \mathbb{E}_{\boldsymbol{y}_1 \sim q(\boldsymbol{y}_1 \mid \boldsymbol{x}_{1:T})} \left[\int \cdots \int_{\boldsymbol{z}_{2:T}} q(\boldsymbol{z}_{2:T} \mid \boldsymbol{x}_{1:T}, \boldsymbol{y}_1) \log \frac{p(\boldsymbol{z}_{2:T} \mid \boldsymbol{y}_1)}{q(\boldsymbol{z}_{2:T} \mid \boldsymbol{x}_{1:T}, \boldsymbol{y}_1)} \, \mathrm{d}\boldsymbol{z}_{2:T} \right]
\end{aligned} \tag{B.5}
$$

$$
\begin{aligned}
&= \mathbb{E}_{(\boldsymbol{z}_{2:T}, \boldsymbol{y}_{1:T}) \sim q_{Z,Y}} \sum_{t=1}^{T} \log p(\boldsymbol{x}_t \mid \boldsymbol{y}_t, \boldsymbol{w}) - D_{\mathrm{KL}}\big(q(\boldsymbol{y}_1 \mid \boldsymbol{x}_{1:k}) \,\big\|\, p(\boldsymbol{y}_1)\big) \\
&\qquad + \mathbb{E}_{\boldsymbol{y}_1 \sim q(\boldsymbol{y}_1 \mid \boldsymbol{x}_{1:k})} \left[\int \cdots \int_{\boldsymbol{z}_{2:T}} q(\boldsymbol{z}_{2:T} \mid \boldsymbol{x}_{1:T}, \boldsymbol{y}_1) \log \frac{p(\boldsymbol{z}_{2:T} \mid \boldsymbol{y}_1)}{q(\boldsymbol{z}_{2:T} \mid \boldsymbol{x}_{1:T}, \boldsymbol{y}_1)} \, \mathrm{d}\boldsymbol{z}_{2:T} \right]
\end{aligned} \tag{B.6}
$$

$$
\begin{aligned}
&= \mathbb{E}_{(\boldsymbol{z}_{2:T}, \boldsymbol{y}_{1:T}) \sim q_{Z,Y}} \sum_{t=1}^{T} \log p(\boldsymbol{x}_t \mid \boldsymbol{y}_t, \boldsymbol{w}) - D_{\mathrm{KL}}\big(q(\boldsymbol{y}_1 \mid \boldsymbol{x}_{1:k}) \,\big\|\, p(\boldsymbol{y}_1)\big) \\
&\qquad + \mathbb{E}_{\boldsymbol{y}_1 \sim q(\boldsymbol{y}_1 \mid \boldsymbol{x}_{1:k})} \left[\int \cdots \int_{\boldsymbol{z}_{2:T}} \prod_{t=2}^{T} q(\boldsymbol{z}_t \mid \boldsymbol{x}_{1:t}) \sum_{t=2}^{T} \log \frac{p(\boldsymbol{z}_t \mid \boldsymbol{y}_1, \boldsymbol{z}_{2:t-1})}{q(\boldsymbol{z}_t \mid \boldsymbol{x}_{1:t})} \, \mathrm{d}\boldsymbol{z}_{2:T} \right]
\end{aligned} \tag{B.7}
$$

$$
\begin{aligned}
&= \mathbb{E}_{(\boldsymbol{z}_{2:T}, \boldsymbol{y}_{1:T}) \sim q_{Z,Y}} \sum_{t=1}^{T} \log p(\boldsymbol{x}_t \mid \boldsymbol{y}_t, \boldsymbol{w}) - D_{\mathrm{KL}}\big(q(\boldsymbol{y}_1 \mid \boldsymbol{x}_{1:k}) \,\big\|\, p(\boldsymbol{y}_1)\big) \\
&\qquad - \mathbb{E}_{\boldsymbol{y}_1 \sim q(\boldsymbol{y}_1 \mid \boldsymbol{x}_{1:k})} D_{\mathrm{KL}}\big(q(\boldsymbol{z}_2 \mid \boldsymbol{x}_{1:t}) \,\big\|\, p(\boldsymbol{z}_2 \mid \boldsymbol{y}_1)\big) \\
&\qquad + \mathbb{E}_{\substack{\boldsymbol{y}_1 \sim q(\boldsymbol{y}_1 \mid \boldsymbol{x}_{1:k}) \\ \boldsymbol{z}_2 \sim q(\boldsymbol{z}_2 \mid \boldsymbol{x}_{1:2})}} \left[\int \cdots \int_{\boldsymbol{z}_{3:T}} \prod_{t=3}^{T} q(\boldsymbol{z}_t \mid \boldsymbol{x}_{1:t}) \sum_{t=3}^{T} \log \frac{p(\boldsymbol{z}_t \mid \boldsymbol{y}_1, \boldsymbol{z}_{2:t-1})}{q(\boldsymbol{z}_t \mid \boldsymbol{x}_{1:t})} \, \mathrm{d}\boldsymbol{z}_{3:T} \right],
\end{aligned} \tag{B.8}
$$

(B.9)

where:

- Equation (B.6) follows from the inference model of Equation (4.8), where $\boldsymbol{y}_1$ only depends on $\boldsymbol{x}_{1:k}$;
- Equation (B.7) is obtained from the factorizations of Equations (4.5), (4.6) and (4.8).

By iterating Equation (B.8)'s step on $\boldsymbol{z}_3, \dots, \boldsymbol{z}_T$ and factorizing all expectations, we obtain:

$$\begin{aligned}
&\log p(\boldsymbol{x}_{1:T} \mid \boldsymbol{w}) \\
&\quad \geq \mathbb{E}_{(\boldsymbol{z}_{2:T}, \boldsymbol{y}_{1:T}) \sim q_{Z,Y}} \sum_{t=1}^{T} \log p(\boldsymbol{x}_t \mid \boldsymbol{y}_t, \boldsymbol{w}) - D_{\mathrm{KL}}\big(q(\boldsymbol{y}_1 \mid \boldsymbol{x}_{1:k}) \,\big\|\, p(\boldsymbol{y}_1)\big) \\
&\qquad - \mathbb{E}_{\boldsymbol{y}_1 \sim q(\boldsymbol{y}_1 \mid \boldsymbol{x}_c)} \Big(\mathbb{E}_{\boldsymbol{z}_t \sim q(\boldsymbol{z}_t \mid \boldsymbol{x}_{1:t})} \Big)_{t=2}^{T} \sum_{t=2}^{T} D_{\mathrm{KL}}\big(q(\boldsymbol{z}_t \mid \boldsymbol{x}_{1:t}) \,\big\|\, p(\boldsymbol{z}_t \mid \boldsymbol{y}_1, \boldsymbol{z}_{1:t-1})\big),
\end{aligned} \tag{B.10}$$

and we finally retrieve Equation (4.9) by using the factorization of Equation (4.8):

$$\begin{aligned}
&\log p(\boldsymbol{x}_{1:T} \mid \boldsymbol{w}) \\
&\quad \geq \mathbb{E}_{(\boldsymbol{z}_{2:T}, \boldsymbol{y}_{1:T}) \sim q_{Z,Y}} \sum_{t=1}^{T} \log p(\boldsymbol{x}_t \mid \boldsymbol{y}_t, \boldsymbol{w}) - D_{\mathrm{KL}}\big(q(\boldsymbol{y}_1 \mid \boldsymbol{x}_{1:k}) \,\big\|\, p(\boldsymbol{y}_1)\big) \\
&\qquad - \mathbb{E}_{(\boldsymbol{z}_{2:T}, \boldsymbol{y}_{1:T}) \sim q_{Z,Y}} \sum_{t=2}^{T} D_{\mathrm{KL}}\big(q(\boldsymbol{z}_t \mid \boldsymbol{x}_{1:t}) \,\big\|\, p(\boldsymbol{z}_t \mid \boldsymbol{y}_{t-1})\big).
\end{aligned} \tag{B.11}$$

B.2. Datasets Details

We detail in this section the datasets used in our experimental study.

B.2.1. Data Representation

For all datasets, video frames are represented by greyscale or RGB pixels with values within $[0, 1]$ obtained by dividing by 255 their original values lying in $[\![0, 255]\!]$.

B.2.2. Stochastic Moving MNIST

This monochrome dataset consists in one or two training MNIST digits (LeCun, Bottou, et al., 1998) of size 28×28 moving linearly within a 64×64 frame and randomly bouncing against its border, sampling a new direction and velocity at each bounce (Denton and Fergus, 2018). We use the same settings as Denton and Fergus (2018):

we train all models on 15 time steps, condition them at testing time on 5 frames, and predict either 20 (for the stochastic version) or 95 (for the deterministic version) frames. Note that we have adapted the dataset to sample more coherent bounces: the original dataset computes digit trajectories that are dependent on the chosen framerate, unlike our corrected version of the dataset. We consequently retrain SVG on this dataset, obtaining comparable results as those originally presented by Denton and Fergus (2018). Testing data are produced by generating a trajectory for each testing digit, and randomly pairwise combining these trajectories to produce 5000 testing sequences, each containing two digits.

B.2.3. KTH Action Dataset (KTH)

This dataset is composed of real-world 64×64 monochrome videos of 25 people performing one of six actions (walking, jogging, running, boxing, handwaving and handclapping) in front of different backgrounds (Schüldt, Laptev, and Caputo, 2004). Uncertainty lies in the appearance of subjects, the action they perform and how it is performed. We use the same settings as Denton and Fergus (2018): we train all models on 20 time steps, condition them at testing time on 10 frames, and predict 30 frames. The training set is formed with actions from the first 20 subjects, the remaining five being used for testing. Training is performed by sampling sub-sequences of size 20 in the training set. The test set is composed of 1000 randomly sampled sub-sequences of size 40.

B.2.4. Human3.6M

This dataset is also made of videos of subjects performing various actions (Ionescu, F. Li, and Sminchisescu, 2011; Ionescu, Papava, et al., 2014). While there are more actions and details to capture with fewer training subjects than in KTH, the video backgrounds are less varied, and subjects always remain within the frames. We use the same settings as Minderer et al. (2019) to train both our model and StructVRNN, for which there is no available pretrained model. We train all models on 16 time steps, condition them at testing time on 8 frames, and predict 45 frames. Videos used in our experiment are subsampled from the original videos at 6.25Hz, center-cropped from 1000×1000 to 800×800 and resized to 64×64 using the Lanczos filter of the Pillow library.[1] The training set is composed of videos of subjects 1, 5, 6, 7, and 8, and the testing set is made from subjects 9 and 11; videos showing more than one action, marked by "`ALL`" in the dataset, are excluded. Training is performed by sampling sub-sequences of size 16 in the training set. The test set is composed of 1000 randomly sampled sub-sequences of size 53 from the testing videos.

B.2.5. BAIR Robot Pushing Dataset (BAIR)

This dataset contains 64×64 videos of a Sawyer robotic arm pushing objects on a tabletop (Ebert et al., 2017). It is highly stochastic as the arm can change its direction

[1] `https://pillow.readthedocs.io/`

at any moment. We use the same settings as Denton and Fergus (2018), train all models on 12 time steps, condition them at testing time on 2 frames, and predict 28 frames. Training and testing sets are the same as those used by Denton and Fergus (2018).

B.3. Training Details

We expose in this section further information needed for the reproducibility of our results.

B.3.1. Architecture

Encoder and decoder architecture. Both g_θ and e_ϕ are chosen to have the same mirrored architecture that depends on the dataset. We use the same architectures as Denton and Fergus (2018): a DCGAN discriminator and generator architecture (Radford, Metz, and Chintala, 2016) for Moving MNIST, and a VGG16 architecture (Simonyan and Zisserman, 2015) – mirrored for e_ϕ – for the other datasets. In both cases, the output of e_ϕ (i.e. $\widetilde{\boldsymbol{x}}$) is a vector of size 128, and g_θ and e_ϕ weights are initialized using a centered Gaussian distribution with a standard deviation of 0.02 (except for biases initialized to 0, and batch normalization layers weights drawn from a Gaussian distribution with unit mean and a standard deviation of 0.02). Additionally, we supplement g_θ with a last sigmoid activation in order to ensure that its outputs lie within $[0, 1]$ like the ground truth data.

Note that, during testing, predicted frames are directly generated by $g_\theta(\boldsymbol{y}_t, \boldsymbol{w})$ without sampling from the observation probability distribution:

$$\mathcal{G}\big(g_\theta(\boldsymbol{y}_t, \boldsymbol{w})\big) = \mathcal{N}\big(g_\theta(\boldsymbol{y}_t, \boldsymbol{w}), \nu I_n\big). \tag{B.12}$$

This is a common practice for Gaussian decoders in VAEs that is adopted by our competitors (A. X. Lee, R. Zhang, et al., 2018; Denton and Fergus, 2018; Minderer et al., 2019).

Content variable. For the Moving MNIST dataset, the content variable $\boldsymbol{w}$ is obtained directly from $\widetilde{\boldsymbol{x}}$ and is a vector of size 128. For KTH, Human3.6M, and BAIR, we supplement this vectorial variable with skip connections from all layers of the encoder g_θ that are then fed to the decoder e_ϕ to handle complex backgrounds. For Moving MNIST, the number of frames k used to compute the content variable is 5; for KTH and Human3.6M, it is 3; for BAIR, it is 2.

The vectorial content variable $\boldsymbol{w}$ is computed from k input frames:

$$\boldsymbol{x}_{\mathrm{c}}^{(k)} = \big(\boldsymbol{x}_{i_1}, \dots, \boldsymbol{x}_{i_k}\big), \tag{B.13}$$

using c_ψ defined as follows:

$$\boldsymbol{w} = c_\psi\Big(\boldsymbol{x}_{\mathrm{c}}^{(k)}\Big) = c_\psi^{(2)}\left(\sum_{j=1}^{k} c_\psi^{(1)}\big(\widetilde{\boldsymbol{x}}_{i_j}\big)\right). \tag{B.14}$$

In other words, c_ψ transforms each frame representation using $c_\psi^{(1)}$, sums these transformations and outputs the application of $c_\psi^{(2)}$ to this sum. Since frame representations $\widetilde{\boldsymbol{x}}_{i_j} = e_\phi\big(\boldsymbol{x}_{i_j}\big)$ are computed independently of each other, c_ψ is indeed permutation-invariant. In practice, $c_\psi^{(1)}$ consists in a linear layer of output size 256 followed by a rectified linear unit (ReLU) activation, while $c_\psi^{(2)}$ is a linear layer of output size 256 (making $\boldsymbol{w}$ of size 256) followed by a hyperbolic tangent activation.

LSTM architecture. The LSTM used for all datasets has a single layer of LSTM cells with a hidden state size of 256.

MLPs architecture. All MLPs used in inference (with parameters ϕ) have three linear layers with hidden size 256 and ReLU activations. All MLPs used in the forward model (with parameters θ) have four linear layers with hidden size 512 and ReLU activations. Any MLP outputting Gaussian distribution parameters (μ, σ) additionally includes a softplus (Dugas et al., 2001) applied to its output dimensions that are used to obtain σ. Weights of f_θ are orthogonally initialized with a gain of 1.2 for KTH and Human3.6M, and 1.41 for the other datasets (except for biases which are initialized to 0), while the other MLPs are initialized with the default weight initialization of PyTorch.

Sizes of latent variables. The sizes of the latent variables in our model are the following: for Moving MNIST, $\boldsymbol{y}$ and $\boldsymbol{z}$ have size 20; for KTH, Human3.6M, and BAIR, $\boldsymbol{y}$ and $\boldsymbol{z}$ have size 50.

Euler step size Models are trained with $\Delta t = 1$ on Moving MNIST, and with $\Delta t = 1/2$ on the others datasets.

B.3.2. Optimization

Models are trained using the Adam optimizer (Kingma and Ba, 2015) with learning rate 3×10^{-4}, and decay rates $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

Loss function. The batch size is chosen to be 128 for Moving MNIST, 100 for KTH and Human3.6M, and 192 for BAIR. The regularization coefficient λ is always set to 1. Logarithms used in the loss are natural logarithms.

For the Moving MNIST dataset, we follow Higgins, Matthey, et al. (2017) by weighting the KLD terms on $\boldsymbol{z}$ (i.e. the sum of KLDs in Equation (4.9)) with a multiplication factor $\beta = 2$.

Variance of the observation. The variance ν considered in the observation probability distribution $\mathcal{G}\big(g_\theta(\boldsymbol{y}, \boldsymbol{w})\big) = \mathcal{N}\big(g_\theta(\boldsymbol{y}_t, \boldsymbol{w}), \nu I_n\big)$ is chosen as follows:

- for Moving MNIST, $\nu = 1$;
- for KTH and Human3.6M, $\nu = 4 \times 10^{-2}$;

Table B.1.: Numerical results for PSNR, SSIM, and LPIPS on BAIR of our model trained with $\Delta t = 1/2$ and tested with different values of Δt.

Step size Δt	PSNR	SSIM	LPIPS
$\Delta t = 1$	18.95 ± 0.25	0.8139 ± 0.0081	0.0640 ± 0.0036
$\Delta t = 1/2$	19.59 ± 0.27	0.8196 ± 0.0084	0.0574 ± 0.0032
$\Delta t = 1/3$	19.49 ± 0.25	0.8201 ± 0.0082	0.0574 ± 0.0032
$\Delta t = 1/4$	19.45 ± 0.26	0.8196 ± 0.0082	0.0579 ± 0.0032
$\Delta t = 1/5$	19.46 ± 0.26	0.8197 ± 0.0082	0.0584 ± 0.0032

- for BAIR, $\nu = \frac{1}{2}$.

Number of optimization steps. The number of optimization steps for each dataset is the following:

- Stochastic Moving MNIST: 1 000 000 steps, with additional 100 000 steps where the learning rate is linearly decreased to 0;
- Deterministic Moving MNIST: 800 000 steps, with additional 100 000 steps where the learning rate is linearly decreased to 0;
- KTH: 150 000 steps, with additional 50 000 steps where the learning rate is linearly decreased to 0;
- Human3.6M: 325 000 steps, with additional 25 000 steps where the learning rate is linearly decreased to 0;
- BAIR: 1 000 000 steps, with additional 500 000 steps where the learning rate is linearly decreased to 0.

Furthermore, the final models for KTH and Human3.6M are chosen among several checkpoints, computed every 5000 iterations for KTH and 20 000 iterations for Human3.6M, as the ones obtaining the best evaluation PSNR. This evaluation score differs from the test score as we extract from the training set an evaluation set by randomly selecting 5% of the training videos from the training set of each dataset. More precisely, the evaluation PSNR for a checkpoint is computed as the mean best prediction PSNR for 400 (for KTH) or 200 (for Human3.6M) randomly extracted sequences of length 30 (for KTH) or 53 (for Human3.6M) from the videos of the evaluation set.

B.4. Influence of the Euler step size

Table B.1 details the numerical results of our model trained on BAIR with $\Delta t = 1/2$ and tested with different values of Δt. It shows that, when refining the Euler approximation, our model maintains its performances in settings unseen during training.

Table B.2.: Numerical results for PSNR, SSIM, and LPIPS on KTH of our model trained with $\Delta t = 1$ and tested with different values of Δt.

Step size Δt	PSNR	SSIM	LPIPS
$\Delta t = 1$	29.77 ± 0.33	0.8681 ± 0.0046	0.0742 ± 0.0029
$\Delta t = 1/2$	29.18 ± 0.35	0.8539 ± 0.0054	0.0882 ± 0.0040
$\Delta t = 1/3$	29.05 ± 0.36	0.8509 ± 0.0056	0.0924 ± 0.0043
$\Delta t = 1/4$	28.98 ± 0.37	0.8496 ± 0.0057	0.0939 ± 0.0045
$\Delta t = 1/5$	28.95 ± 0.37	0.8490 ± 0.0058	0.0948 ± 0.0045

Table B.3.: Numerical results for PSNR, SSIM, and LPIPS on KTH of our model trained with $\Delta t = 1/2$ and tested with different values of Δt.

Step size Δt	PSNR	SSIM	LPIPS
$\Delta t = 1$	28.80 ± 0.25	0.8495 ± 0.0053	0.0994 ± 0.0044
$\Delta t = 1/2$	29.69 ± 0.32	0.8697 ± 0.0046	0.0736 ± 0.0029
$\Delta t = 1/3$	29.52 ± 0.33	0.8656 ± 0.0048	0.0777 ± 0.0033
$\Delta t = 1/4$	29.43 ± 0.33	0.8633 ± 0.0049	0.0790 ± 0.0034
$\Delta t = 1/5$	29.35 ± 0.34	0.8615 ± 0.0050	0.0811 ± 0.0036

Tables B.2 and B.3 detail the numerical results of our model trained on KTH with, respectively, $\Delta t = 1$ and $\Delta t = 1/2$, and tested with different values of Δt. They show that if Δt is chosen too high when training (here, $\Delta t = 1$), the model performance drops when refining the Euler approximation. We assume that this phenomenon arises because the Euler approximation used in training is too rough, making the model adapt to an overly discretized dynamic that cannot be transferred to smaller Euler step sizes. Indeed, when training with a smaller step size (here, $\Delta t = 1/2$), results in the training settings are equivalent while results obtained with a lower Δt are now much closer, if not equivalent, to the nominal ones. This shows that the model learns a continuous dynamics if learned with a small enough step size.

Note that the loss of performance using a higher Δt in testing than in training, like in Tables B.1 and B.3, is expected as it corresponds to loosening the Euler approximation compared to training. However, even in this challenging setting, our model maintains state-of-the-art results, demonstrating the quality of the learned dynamic as it can be further discretized if needed at the cost of a reasonable drop in performance.

B.5. Pendulum Experiment

We test the ability of our method to model the dynamics of a common dataset used in the literature of state-space models (Karl et al., 2017; Fraccaro, Kamronn, et al., 2017), Pendulum (Karl et al., 2017). It consists of noisy observations of a dynamic

Table B.4.: ELBO, in nats, achieved by DVBF, KVAE and our model on the Pendulum dataset. The bold score indicates the best performing method.

DVBF	KVAE	Ours
798.56	**807.02**	806.12

torque-controlled pendulum; it is stochastic as the information of this control is not available. We test our model, without the content variable $\boldsymbol{w}$, in the same setting as DVBF (Karl et al., 2017) and KVAE (Fraccaro, Kamronn, et al., 2017) and report the corresponding ELBO scores in Table B.4. The encoders and decoders for all methods are MLPs.

Our model outperforms DVBF and is merely beaten by KVAE. This can be explained by the nature of the KVAE model, whose sequential model is learned using a Kalman filter rather than a VAE, allowing exact inference in the latent space. On the contrary, DVBF is learned, like our model, by a sequential VAE, and is thus closer to our model than KVAE. This result then shows that the dynamic model that we propose in the context of sequential VAEs is more adapted on this dataset than the one of DVBF, and achieve results close to a method taking advantage of exact inference using adapted tools such as Kalman filters.

B.6. Additional Samples

This section includes additional samples corresponding to experiments described in Section 4.4.

B.6.1. Stochastic Moving MNIST

We present in Figures B.1 to B.4 additional samples from SVG and our model on Stochastic Moving MNIST.

In particular, Figure B.3 shows SVG changing a digit shape in the course of a prediction even though it does not cross another digit, whereas ours maintain the digit shape. We assume that the advantage of our model comes from the latent nature of its dynamic and the use of a static content variable that is prevented from containing temporal information. Indeed, even when the best sample from our model is not close from the ground truth of the dataset, like in Figure B.4, it still maintains the shapes of the digits.

B.6.2. KTH

We present in Figures B.5 to B.9 additional samples from SV2P, SVG, SAVP and our model on KTH, with additional insights.

B.6.3. Human3.6M

We present in Figures B.10 and B.11 additional samples from StructVRNN and our model on Human3.6M, with additional insights.

B.6.4. BAIR

We present in Figures B.12 to B.14 additional samples from SV2P, SVG, SAVP and our model on BAIR.

B.6.5. Oversampling

We present in Figure B.15 additional examples of video generation at a doubled and quadrupled frame rate by our model.

B.6.6. Content Swap

We present in Figures B.16 to B.20 additional examples of content swap as in Figure 4.9.

B.6.7. Interpolation in the Latent Space

We present in Figures B.21 and B.22 additional examples of interpolation in the latent space between two trajectories as in Figure 4.10.

Figure B.1.: Conditioning frames and corresponding ground truth and best samples with respect to PSNR from SVG and our method, and worst and random samples from our method, for an example of the Stochastic Moving MNIST dataset.

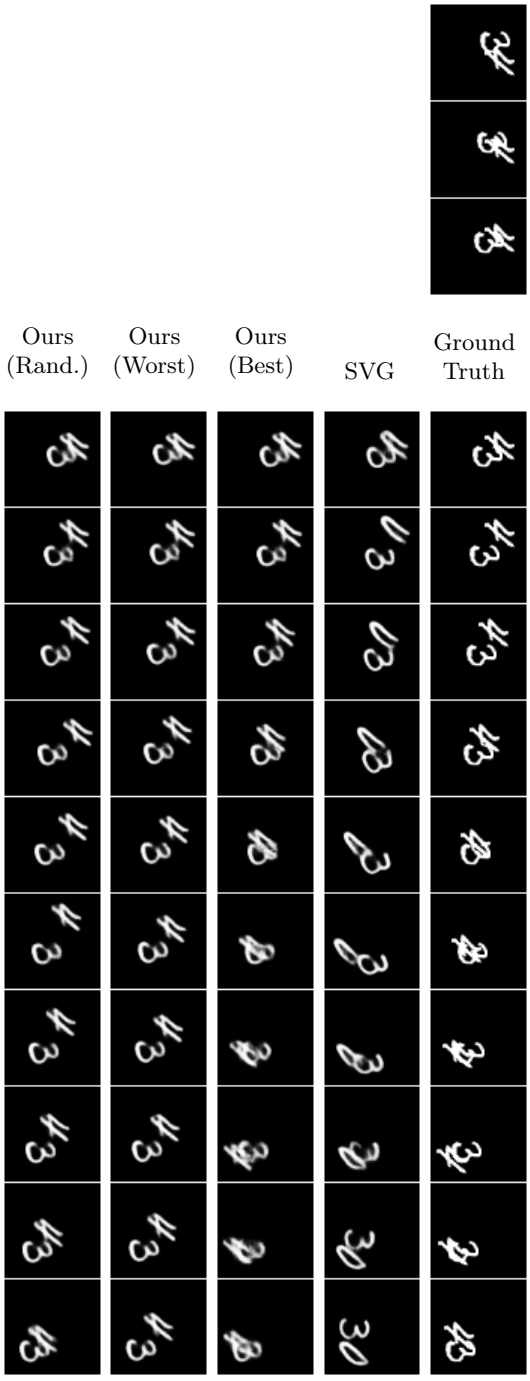


Figure B.2.: Additional samples for the Stochastic Moving MNIST dataset (cf. Figure B.1).

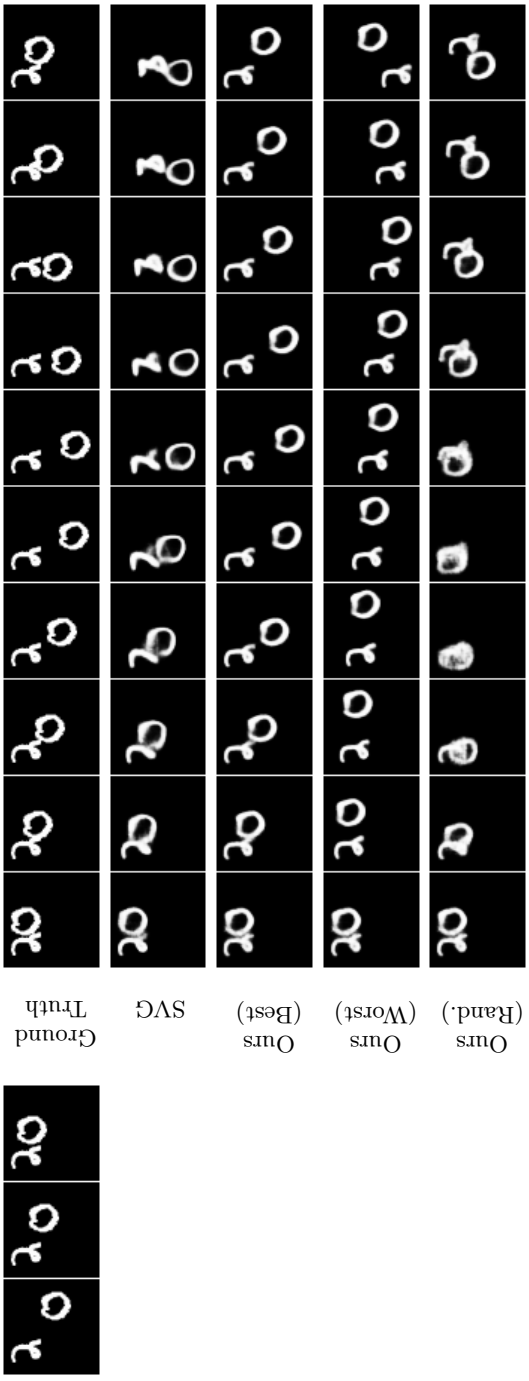


Figure B.3.: Additional samples for the Stochastic Moving MNIST dataset (cf. Figure B.1). SVG fails to maintain the shape of a digit, while ours is temporally coherent.

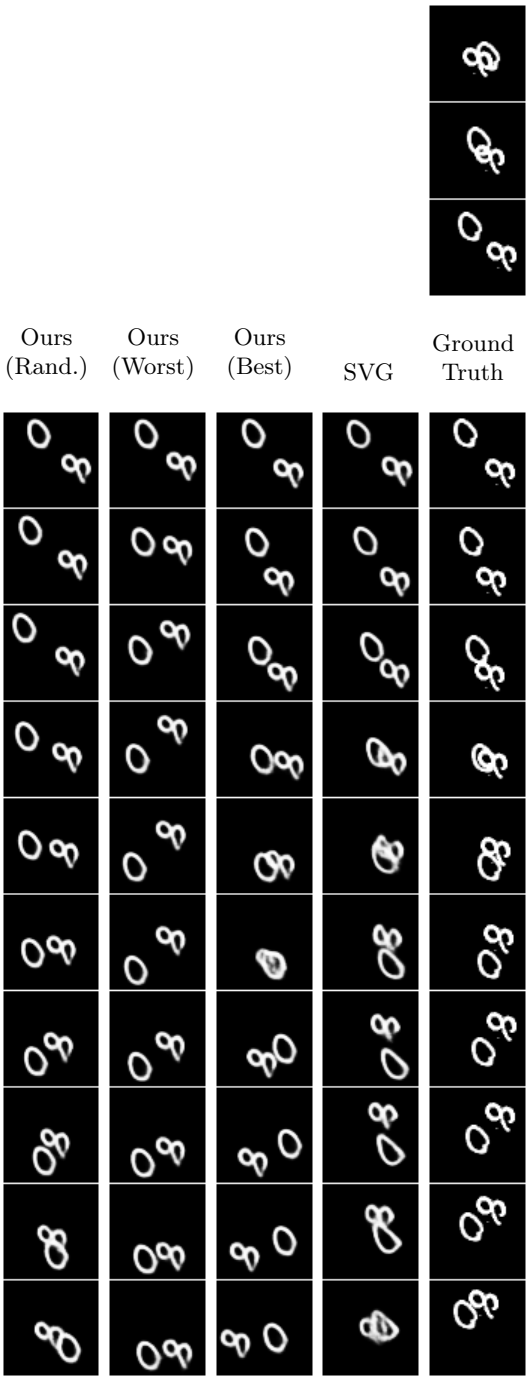


Figure B.4.: Additional samples for the Stochastic Moving MNIST dataset (cf. Figure B.1). This example was chosen in the worst 1% testing examples of our model with respect to PSNR. Despite this adversarial criterion, our model maintains temporal consistency as digits are not deformed in the course of the video.

Figure B.5.: Conditioning frames and corresponding ground truth, best samples from SVG, SAVP and our method, and worst and random samples from our method, for an example of the KTH dataset. Samples are chosen according to their LPIPS with respect to the ground truth. On this specific task (clapping), all methods but SV2P (which produces blurry predictions) perform well, even though ours stays closer to the ground truth.

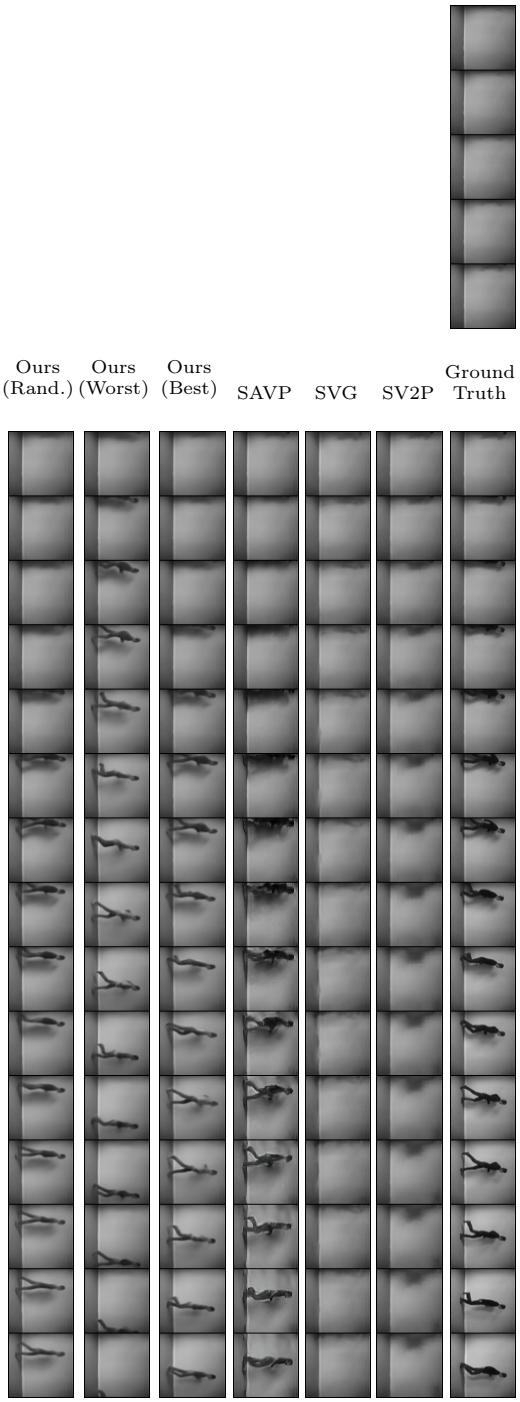


Figure B.6.: Additional samples for the KTH dataset (cf. Figure B.5). In this example, the shadow of the subject is visible in the last conditioning frames, foreshadowing its appearance. This is a failure case for SV2P and SVG which only produce an indistinct shadow, whereas SAVP and our model make the subject appear. Yet, SAVP produces the wrong action and an inconsistent subject in its best sample, while ours is correct.

Figure B.7.: Additional samples for the KTH dataset (cf. Figure B.5). This example is a failure case for all methods: SV2P produces blurry frames, SVG and SAVP are not consistent (change of action or subject appearance in the video), and our model produces a ghost image at the end of the prediction on the worst sample only.

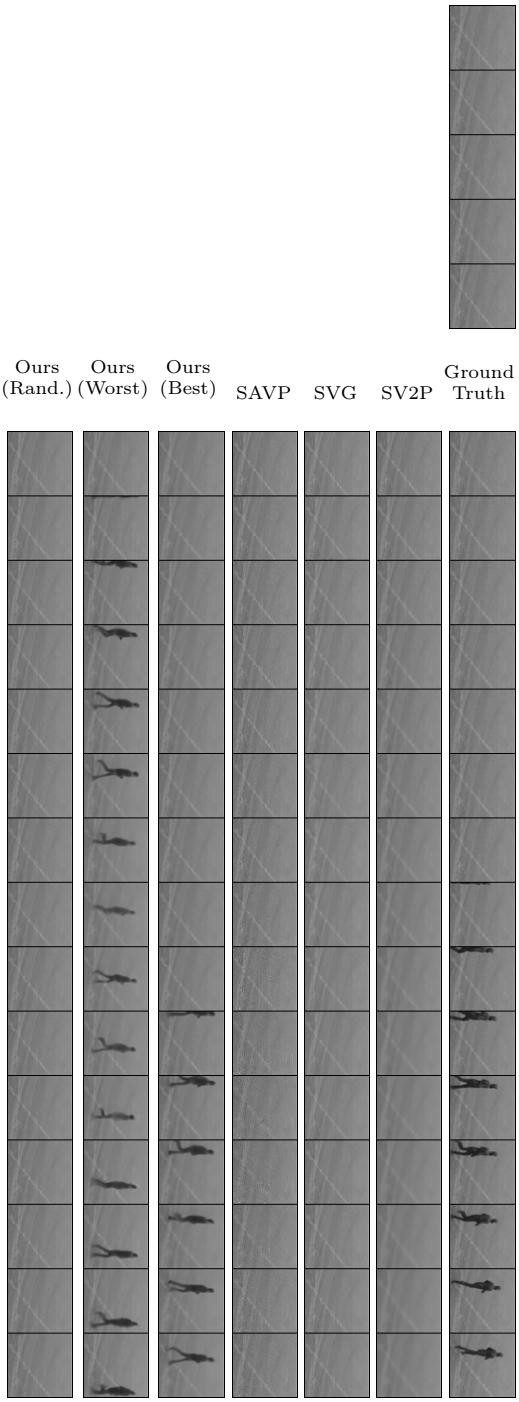


Figure B.8.: Additional samples for the KTH dataset (cf. Figure B.5). Our model is the only one to make a subject appear, like in the ground truth.

Figure B.9.: Additional samples for the KTH dataset (cf. Figure B.5). The subject in this example is boxing, which is a challenging action in the dataset as all methods are far from the ground truth, even though ours remain closer in this case as well.

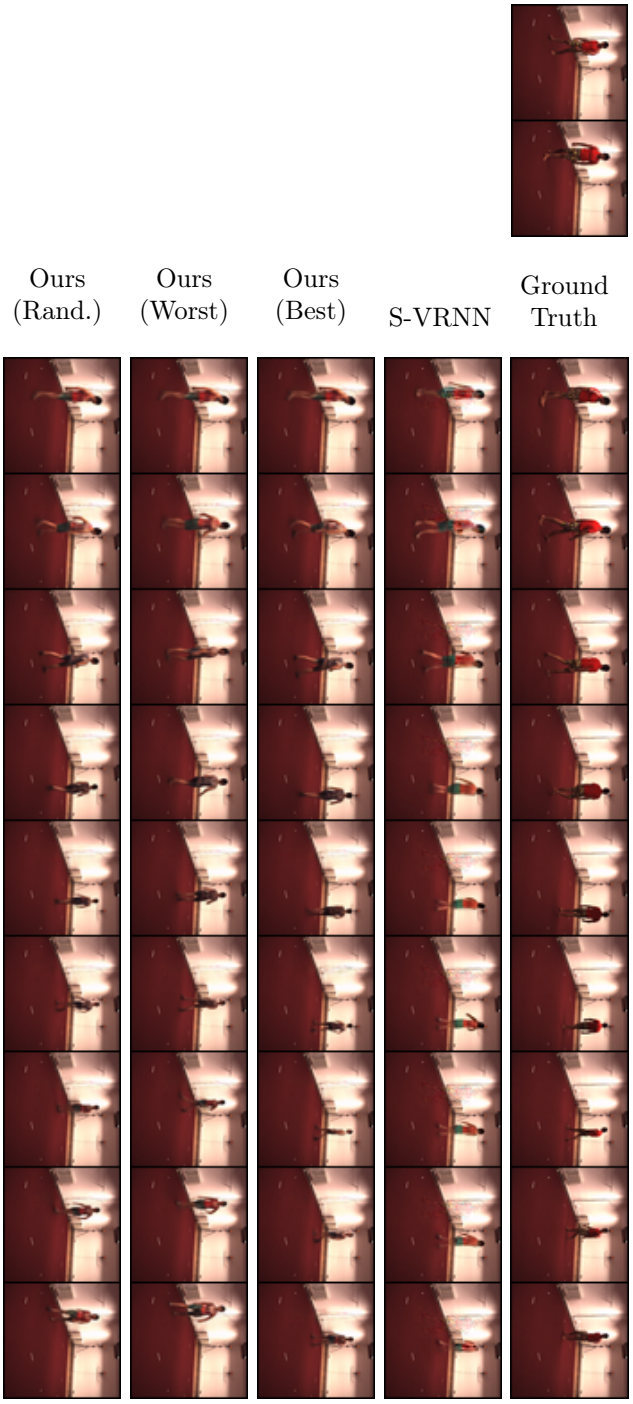


Figure B.10.: Conditioning frames and corresponding ground truth, best samples from StructVRNN and our method, and worst and random samples from our method, for an example of the Human3.6M dataset. Samples are chosen according to their LPIPS with respect to the ground truth. We better capture the movements of the subject as well as their diversity, predict more realistic subjects, and present frames with less artefacts.

Figure B.11.: Additional samples for the Human3.6M dataset (cf. Figure B.10). This action is better captured by our model, which is able to produce diverse realistic predictions.

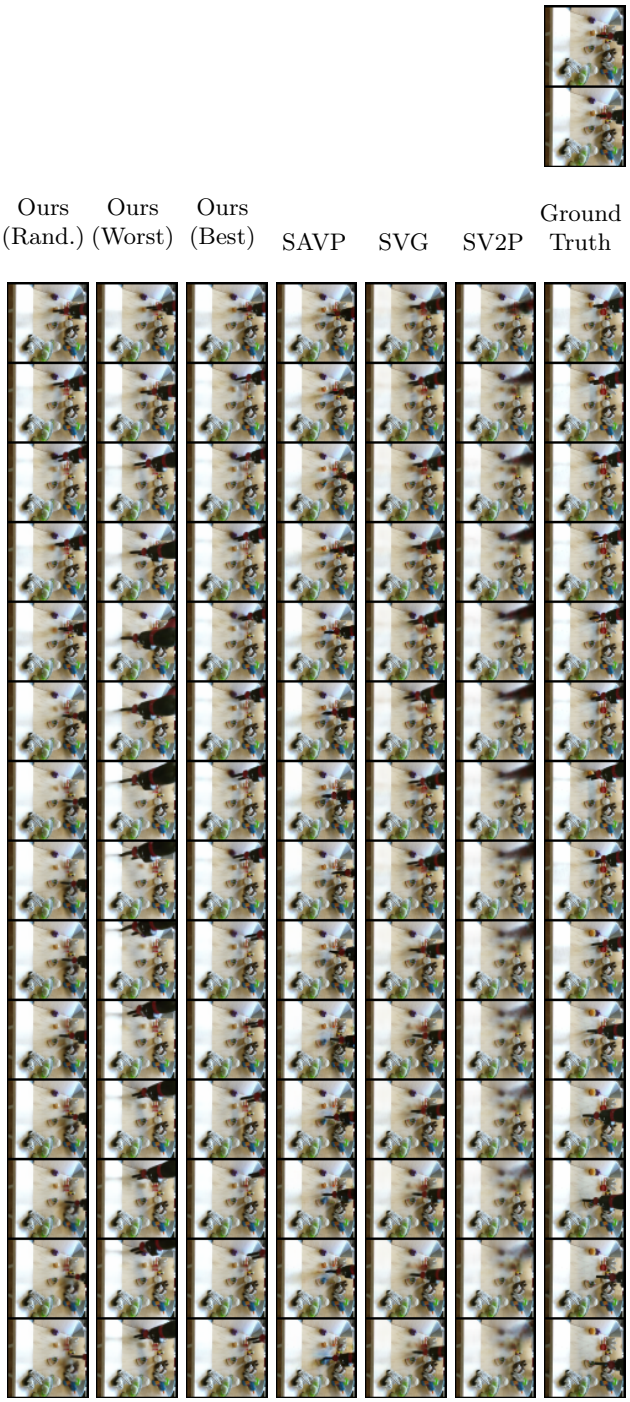


Figure B.12.: Conditioning frames and corresponding ground truth, best samples from SVG, SAVP and our method, and worst and random samples from our method, for an example of the BAIR dataset. Samples are chosen according to their LPIPS with respect to the ground truth.

Figure B.13.: Additional samples for the BAIR dataset (cf. Figure B.12).

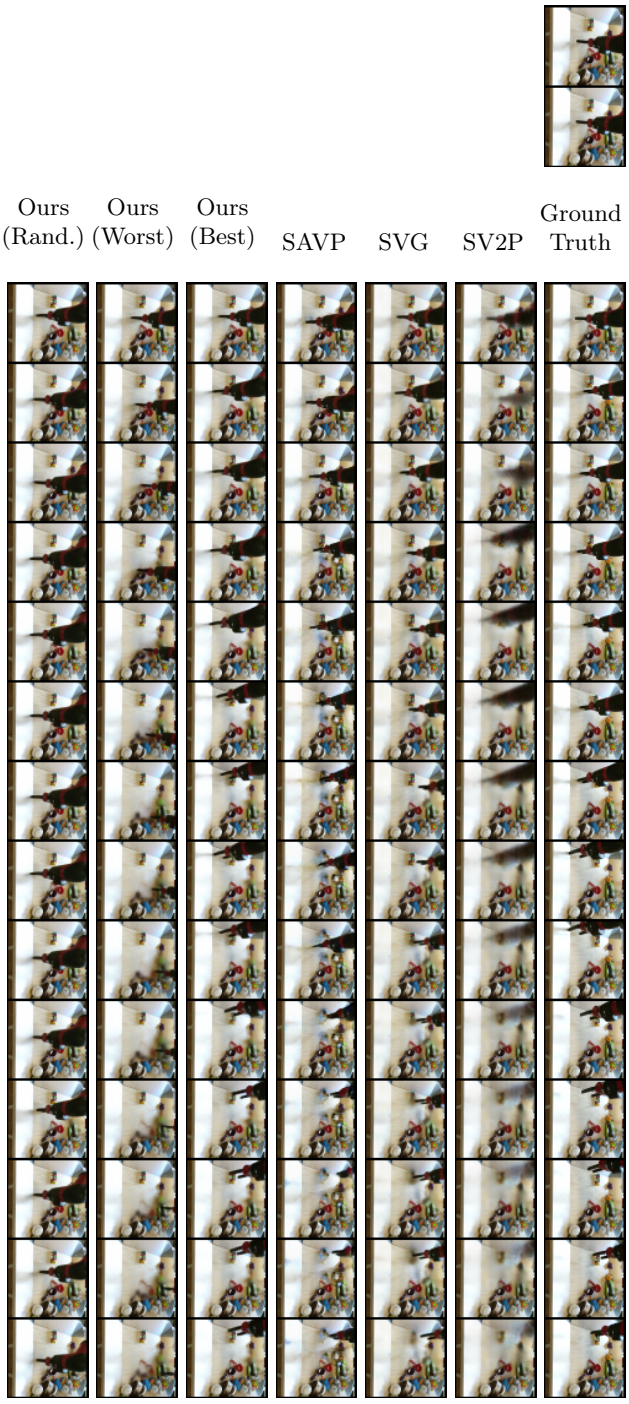
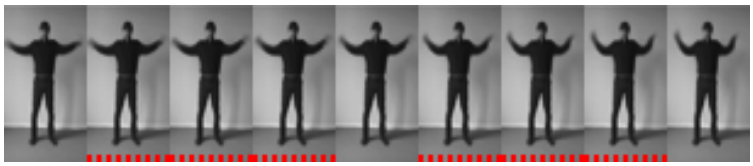
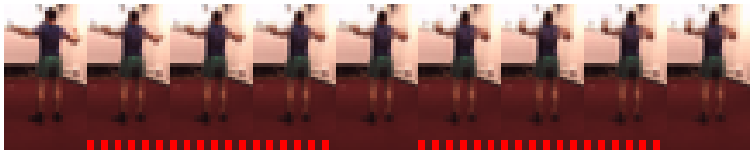


Figure B.14.: Additional samples for the BAIR dataset (cf. Figure B.12).



(a) Cropped KTH sample, centered on the subject (training $\Delta t = 1/2$).



(b) Cropped Human3.6M sample, centered on the subject (training $\Delta t = 1/2$).

(c) Cropped BAIR sample, centered on the robot arm (training $\Delta t = 1$).

Figure B.15.: Generation examples at doubled or quadrupled frame rate, using a halved Δt compared to training. Frames including a bottom red dashed bar are intermediate frames.

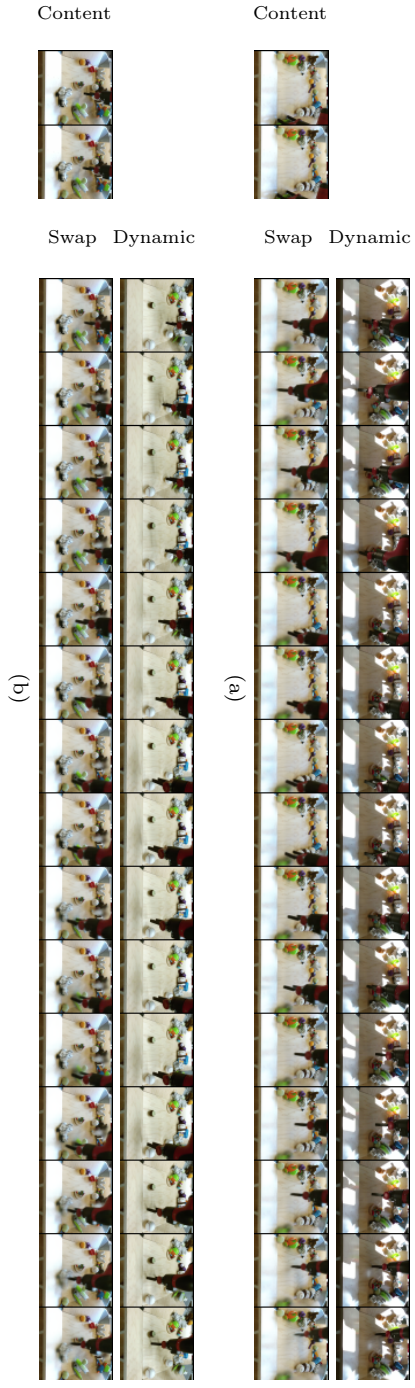


Figure B.16.: (a), (b) Video (bottom right) generated from the combination of dynamic variables ($\boldsymbol{y}$, $\boldsymbol{z}$) inferred with a video (top) and a content variable ($\boldsymbol{w}$) computed with the conditioning frames of another video (bottom left).

Figure B.17.: Additional example of content swap (cf. Figure B.16). In this example, the extracted content is the video background, which is successfully transferred to the target video.

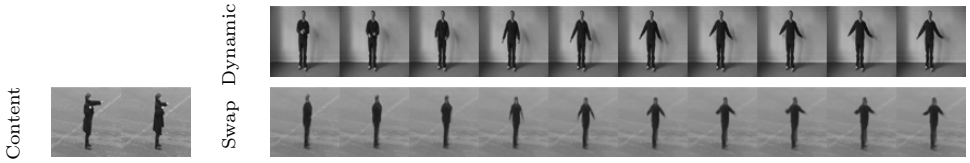


Figure B.18.: Additional example of content swap (cf. Figure B.16). In this example, the extracted content is the video background and the subject appearance, which are successfully transferred to the target video.

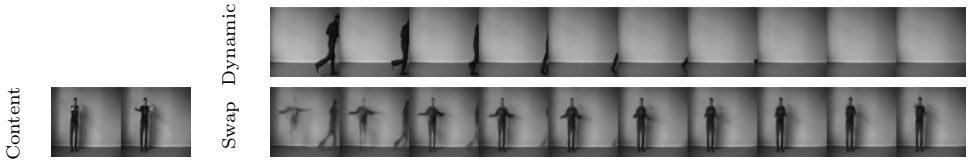


Figure B.19.: Additional example of content swap (cf. Figure B.16). This example shows a failure case of content swapping.

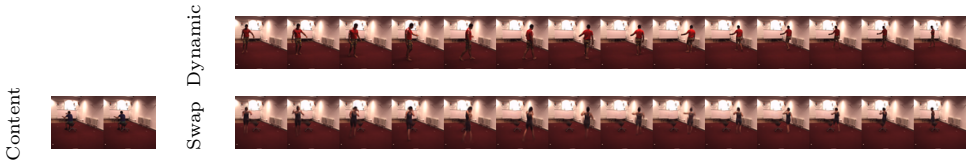
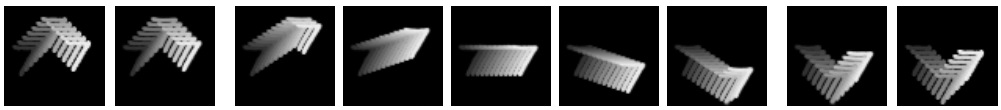


Figure B.20.: Additional example of content swap (cf. Figure B.16).



(a) Ref. 1 (b) Rec. 1 (c) Interpolation (d) Rec. 2 (e) Ref. 2

Figure B.21.: From left to right, $\boldsymbol{x}^{\mathrm{s}}$, $\widehat{\boldsymbol{x}}^{\mathrm{s}}$ (reconstruction of $\boldsymbol{x}^{\mathrm{s}}$ by the VAE of our model), results of the interpolation in the latent space between $\boldsymbol{x}^{\mathrm{s}}$ and $\boldsymbol{x}^{\mathrm{t}}$, $\widehat{\boldsymbol{x}}^{\mathrm{t}}$ and $\boldsymbol{x}^{\mathrm{t}}$. Each trajectory is materialized in shades of grey in the frames.

(a) Ref. 1 (b) Rec. 1 (c) Interpolation (d) Rec. 2 (e) Ref. 2

Figure B.22.: Additional example of interpolation in the latent space between two trajectories (cf. Figure B.21).

Appendix C.

Supplementary Material of Chapter 5

In this appendix chapter of Chapter 5, we provide supplementary details, explanations and discussions as well as additional experiments. Appendix C.1 details and proves the resolution of PDEs mentioned in Section 5.2. Appendix C.2 further discusses the spatial constraint of Equation (5.14). Appendix C.3 justifies in detail the introduction of the penalty on the initial condition of Equation (5.15) based on our differential equations point of view. Appendix C.4 presents the datasets leveraged in Section 5.4 and associated technical details. Appendix C.5 provides the full training details and hyperparameter choices required to reproduce our results. Finally, Appendix C.6 contains new experimental results as well as additional prediction samples for all datasets and compared methods in Section 5.4.

C.1. Proofs

C.1.1. Resolution of the Heat Equation

In this section, we succinctly detail a proof for the existence and uniqueness of the solution to the two-dimensional heat equation. It shows that product-separable solutions allow building the entire solution space for this problem, highlighting our interest in the research of separable solutions.

Existence through separation of variables. Consider the heat equation problem:

$$\frac{\partial u}{\partial t} = c^2 \frac{\partial^2 u}{\partial p^2}, \qquad u(0,t) = u(L,t) = 0, \qquad u(p,0) = f(p). \tag{C.1}$$

Assuming product separability of u with $u(p,t) = u_1(p)u_2(t)$ in Equation (C.1) gives:

$$c^2 \frac{u_1''(p)}{u_1(p)} = \frac{u_2'(t)}{u_2(t)}. \tag{C.2}$$

Both sides being independent of each other variables, they are equal to a constant denoted by $-\alpha$. If α is negative, solving the right side of Equation (C.2) results to non-physical solutions with exponentially increasing temperatures, and imposing border condition of Equation (C.1) makes this solution collapse to the null trivial solution. Therefore, we consider that $\alpha > 0$.

Both sides of Equation (C.2) being equal to a constant leads to a second-order ODE on u_1 and a first-order ODE on u_2, giving the solution shapes, with constants A, B and D:

$$\begin{cases} u_1(p) &= A\cos\left(\sqrt{\alpha}p\right) + B\sin\left(\sqrt{\alpha}p\right) \\ u_2(t) &= D\mathrm{e}^{-\alpha c^2 t} \end{cases}. \tag{C.3}$$

Link with initial and boundary conditions. We now link the above equation to the boundary conditions of the problem. Because our separation is multiplicative, we can omit D for non-trivial solutions and set it without loss of generality to 1, as it only scales the values of A and B.

Boundary conditions $u(0,t) = u(L,t) = 0$, along with the fact that for all $t > 0$, $u_2(t) \neq 0$, give:

$$A = 0, \qquad B\mathrm{e}^{-\alpha c^2 t}\sin\left(\sqrt{\alpha}L\right) = 0, \tag{C.4}$$

which means that, for a non-trivial solution (i.e., $B \neq 0$), we have for some $n \in \mathbb{N}$: $\sqrt{\alpha} = {}^{n\pi}/_{L}$. We can finally express our product-separable solution to the heat equation without initial conditions as:

$$u(p,t) = B\sin\left(\frac{n\pi}{L}p\right)\exp\left(-\left(\frac{cn\pi}{L}\right)^2 t\right). \tag{C.5}$$

Considering the superposition principle, because the initial problem is homogeneous, all linear combinations of Equation (C.5) are solutions of the heat equation without initial conditions. Therefore, any following function is a solution of the heat equation without initial conditions.

$$u(p,t) = \sum_{n=0}^{+\infty} B_n \sin\left(\frac{n\pi}{L}p\right)\exp\left(-\left(\frac{cn\pi}{L}\right)^2 t\right). \tag{C.6}$$

Finally, considering the initial condition $u(p,0) = f(p)$, a Fourier decomposition of f allows to choose appropriate values for all coefficients B_n, showing that, for any initial condition f, there exists a solution to Equation (C.1) of the form of Equation (C.6).

Uniqueness. We present here elements of proof to establish the uniqueness of the solutions of Equation (C.1) that belong to $\mathcal{C}^2\left([0,1]\times\mathbb{R}_+\right)$. Detailed and rigorous proofs are given by Le Dret and Lucquin (2016).

The key element consists in establishing the so-called Maximum Principle which states that, considering a sufficiently smooth solution, the minimum value of the solution is reached on the boundary of the space and time domains.

For null border conditions as in our case, this means that the norm of the solution u is given by the norm of the initial condition f. Finally, let us consider two smooth solutions U_1 and U_2 of Equation (C.1). Then, their difference $v = U_1 - U_2$ follows the heat equation with null border and initial conditions (i.e, $v(p,0) = 0$). Because v is as regular as U_1 and U_2, it satisfies the previous fact about the norm of the solutions, i.e.

the norm of v equals the norm of its initial condition: $\|v\| = 0$. Therefore, v is null and so is $U_1 - U_2 = 0$, showing the uniqueness of the solutions.

This shows that solutions of the form of Equation (C.6) shape the whole set of smooth solutions of Equation (C.1).

C.1.2. Heat Equation with Advection Term

Consider the heat equation with a complementary advection term, for $p \in (-1, 1)$, $t \in (0, T)$ and a constant $c \in \mathbb{R}_+$.

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial p} = \chi \frac{\partial^2 u}{\partial p^2}. \tag{C.7}$$

We give here details for the existence of product-separable solutions of Equation (C.7). To this end, let us choose real constants α and β, and consider the following change of variables for u:

$$u(p, t) = v(p, t)\mathrm{e}^{\alpha p + \beta t}. \tag{C.8}$$

The partial derivatives from Equation (C.7) can be rewritten as functions of the new variable v:

$$\frac{\partial u}{\partial t} = \frac{\partial v}{\partial t}\,\mathrm{e}^{\alpha p + \beta t} + \beta v \mathrm{e}^{\alpha p + \beta t}, \tag{C.9}$$

$$\frac{\partial u}{\partial p} = \frac{\partial v}{\partial p}\,\mathrm{e}^{\alpha p + \beta t} + \alpha v \mathrm{e}^{\alpha p + \beta t}, \tag{C.10}$$

$$\frac{\partial^2 u}{\partial p^2} = \frac{\partial^2 v}{\partial p^2}\,\mathrm{e}^{\alpha p + \beta t} + 2\alpha\,\frac{\partial v}{\partial p}\,\mathrm{e}^{\alpha p + \beta t} + \alpha^2 v \mathrm{e}^{\alpha p + \beta t}. \tag{C.11}$$

Using these expressions in Equation (C.7) and dividing it by $\mathrm{e}^{\alpha p + \beta t}$ lead to:

$$\frac{\partial v}{\partial t} + \Big(\beta + c\alpha - \alpha^2\chi\Big)v + (c - 2\alpha\chi)\frac{\partial v}{\partial p} = \nu \frac{\partial^2 v}{\partial p^2}. \tag{C.12}$$

α and β can then be set such that:

$$\beta + c\alpha - \alpha^2\chi = 0, \qquad\qquad c - 2\alpha\chi = 0, \tag{C.13}$$

to retrieve the standard two-dimensional heat equation of Equation (C.1) given by:

$$\frac{\partial v}{\partial t} = \chi \frac{\partial^2 v}{\partial p^2}, \tag{C.14}$$

which is known to have product-separable solutions as explained in the previous section. This more generally shows that all solutions of Equation (C.7) can be retrieved from solutions to Equation (C.1).

C.2. Accessing Time Derivatives of w and Deriving a Feasible Weaker Constraint

Explicitly constraining the time derivative of $e_\theta^S(\boldsymbol{x}_{t::\tau})$ as explained in Section 5.3.4 is a difficult matter in practice. Indeed, e_θ^S takes as input neither the time coordinate t nor spatial coordinates like abscissa and ordinate as done by Raissi (2018) and Sirignano and Spiliopoulos (2018), which allows them to directly estimate the networks' derivatives with respect to these coordinates thanks to automatic differentiation. In our case, e_θ^S rather takes as input a finite number of observations, making this derivative impractical to compute.

To discretize Equation (5.13) and find a weaker constraint, we choose to leverage the Cauchy-Schwarz inequality. We present and use in Chapter 5 a version where we apply this inequality on the whole integration domain, i.e. from t_0 to $t_1 - \tau\Delta t$. We highlight that this inequality can also be applied on subintervals of the integration domain, generalizing our proposition. Indeed, let $p \in \mathbb{N}^*$ and consider a sequence of $t^{(k)}$ for $k \in [\![0, p]\!]$ such that $t_0 = t^{(0)} \leq t^{(1)} \leq \ldots \leq t^{(p)} = t_1 - \tau\Delta t$. Then, using the Cauchy-Schwarz inequality, we obtain:

$$\begin{aligned}
\int_{t_0}^{t_1-\tau\Delta t} \left\| \frac{\partial e_\theta^S(\boldsymbol{x}_{t::\tau})}{\partial t} \right\|_2^2 \mathrm{d}t &= \sum_{k=0}^{k=p} \int_{t^{(k-1)}}^{t^{(k)}} \left\| \frac{\partial e_\theta^S(\boldsymbol{x}_{t::\tau})}{\partial t} \right\|_2^2 \mathrm{d}t \\
&\geq \sum_{k=0}^{k=p} \frac{1}{t^{(k)} - t^{(k-1)}} \left\| \int_{t^{(k-1)}}^{t^{(k)}} \frac{\partial e_\theta^S(\boldsymbol{x}_{t::\tau})}{\partial t} \mathrm{d}t \right\|_2^2 \\
&\geq \sum_{k=0}^{k=p} \frac{1}{t^{(k)} - t^{(k-1)}} \left\| e_\theta^S(\boldsymbol{x}_{t^{(k)}::\tau}) - e_\theta^S(\boldsymbol{x}_{t^{(k-1)}::\tau}) \right\|_2^2.
\end{aligned} \tag{C.15}$$

Our constraint is a special case of this development, with $p = 1$. Nevertheless, we experimentally find that our simple penalty is sufficient to achieve state-of-the-art performance at a substantially reduced computational cost. We notice that other invariance constraints such as the one of Denton and Birodkar (2017) can also be derived thanks to this framework, showing the generality of our approach.

C.3. Of Spatiotemporal Disentanglement

C.3.1. Separation of Variables Preserves the Mutual Information of w and y through Time

C.3.1.1. Invertible Flow of an ODE

As argued in Section 2.1.2.1, the Cauchy-Lipschitz theorem ensures the existence and uniqueness of a solution to the ODE of Equation (5.10) given an initial condition.

Consequently, the flow of this ODE, denoted by Φ_t and defined as:

$$\begin{aligned}\Phi\colon \mathbb{R}\times\mathbb{R}^p &\to \mathbb{R}^p\\ (t_0, \boldsymbol{y}_{t_0}) &\mapsto \Phi_t(\boldsymbol{y}_{t_0}) = \boldsymbol{y}_{t_0+t}\end{aligned}$$

is a bijection for all t. Indeed, let $\boldsymbol{y}_{t_0}$ be fixed and t_0, t_1 be two time steps. Thanks to the existence and uniqueness of the solution to the ODE with this initial condition, the following holds:

$$\Phi_{t_0+t_1} = \Phi_{t_0}\circ\Phi_{t_1} = \Phi_{t_1}\circ\Phi_{t_0}. \tag{C.16}$$

Therefore, Φ_t is a bijection and $\Phi_t^{-1} = \Phi_{-t}$. Moreover, the flow is differentiable if f_θ is continuously differentiable as well, which is not a restrictive assumption if it is implemented by a neural network with differentiable activation functions.

C.3.1.2. Preservation of Mutual Information by Invertible Mappings

To justify the introduction of Equation (5.15), we leverage the result stating that invertible transformations of random variables preserve their mutual information. A proof of this result is given by Kraskov, Stögbauer, and Grassberger (2004).

We indicate below the major steps of the proof. Let X and Y be two random variables with marginal densities μ_X and μ_Y. Let F be a diffeomorphism acting on Y, with $Y' = F(Y)$. If J_F is the determinant of the Jacobian of F, we have:

$$\mu'\big(x, y'\big) = \mu(x, y)\mathrm{J}_F\big(y'\big).$$

Then, expressing the mutual information I in integral form, with the change of variable $y' = F(y)$ (F being a diffeomorphism), results in:

$$\begin{aligned}I\big(X, Y'\big) &= \iint_{x,y'} \mu'\big(x, y'\big)\log\frac{\mu'\big(x, y'\big)}{\mu_X(x)\times\mu_{Y'}(y')}\,\mathrm{d}x\,\mathrm{d}y'\\ &= \iint_{x,y} \mu(x, y)\log\frac{\mu(x, y)}{\mu_X(x)\times\mu_Y(y)}\,\mathrm{d}x\,\mathrm{d}y\\ I\big(X, Y'\big) &= I(X, Y),\end{aligned}$$

which is the desired result.

C.3.2. Ensuring Disentanglement at any Time

As noted by X. Chen et al. (2016) and Achille and Soatto (2018), mutual information I is a key metric to evaluate disentanglement. We show that our model naturally preserves the mutual information between $\boldsymbol{w}$ and $\boldsymbol{y}$ through time thanks to the flow of the learned ODE on $\boldsymbol{y}$. Indeed, with the previous result of mutual information preservation by diffeomorphisms, and Φ_t being a diffeomorphism as demonstrated above, we have, for all t and t':

$$I(\boldsymbol{w}, \boldsymbol{y}_t) = I\big(\boldsymbol{w}, \Phi_{t'-t}(\boldsymbol{y}_t)\big) = I(\boldsymbol{w}, \boldsymbol{y}_{t'}). \tag{C.17}$$

Hence, if $\boldsymbol{w}$ and $\boldsymbol{y}_t$ are disentangled, then so are $\boldsymbol{w}$ and $\boldsymbol{y}_{t'}$.

The flow Φ_t being discretized in practice, its invertibility can no longer be guaranteed in general. Some numerical schemes (Z. Chen et al., 2020) or residual networks with Lipschitz-constrained residual blocks (Behrmann et al., 2019) provide sufficient conditions to concretely reach this invertibility. In our case, we did not observe the need to enforce invertibility. We can also leverage the data processing inequality to show that, for any $t \geq t_0$:

$$I(\boldsymbol{w}, \boldsymbol{y}_{t_0}) \geq I(\boldsymbol{w}, \boldsymbol{y}_t), \tag{C.18}$$

since $\boldsymbol{y}_t$ is a deterministic function of $\boldsymbol{y}_{t_0}$. Since we constrain the very first $\boldsymbol{y}$ value $\boldsymbol{y}_{t_0}$ (i.e. we do not need to go back in time), there is no imperative need to enforce the invertibility of Φ_t in practice: the inequality also implies that, if $\boldsymbol{w}$ and $\boldsymbol{y}_{t_0}$ are disentangled, then so are $\boldsymbol{w}$ and $\boldsymbol{y}_t$ for $t \geq t_0$. Nevertheless, should the need to disentangle for $t < t_0$ appear, the aforementioned mutual information conservation properties could allow, with further practical work to ensure the effective invertibility of Φ_t, to still regularize $\boldsymbol{y}_{t_0}$ only. This is, however, out of the scope of this work.

C.4. Datasets

C.4.1. WaveEq and WaveEq-100

These datasets are based on the two-dimensional wave equation on a functional $w(x, y, t)$:

$$\frac{\partial^2 w}{\partial t^2} = c^2 \nabla^2 w + f(x, y, t), \tag{C.19}$$

where in this context x and y are spatial coordinates, ∇^2 is the Laplacian operator with respect to x and y, c denotes the wave celerity, and f is an arbitrary time-dependent source term. It has several applications in physics, modeling a wide range of phenomena ranging from mechanical oscillations to electromagnetism. Note that the homogeneous equation, where $f = 0$, admits product-separable solutions.

We build the WaveEq dataset by solving Equation (C.19) for $t \in [0, 0.298]$ and $x, y \in [0, 63]$. Sequences are generated using c drawn uniformly at random in $[300, 400]$ for each sequence to imitate the propagation of acoustic waves, with initial and Neumann boundary conditions:

$$w(x, y, 0) = w(0, 0, t) = w(32, 32, t) = 0. \tag{C.20}$$

Following Saha, Dash, and Mukhopadhyay (2020), we make use of the following source term:

$$f(x, y, t) = \begin{cases} f_0 \mathrm{e}^{-\frac{t}{T_0}} & \text{if } (x, y) \in \mathcal{B}\big((32, 32), 5\big) \\ 0 & \text{otherwise} \end{cases}, \tag{C.21}$$

with $T_0 = 0.05$ and $f_0 \sim \mathcal{U}\big([1, 30]\big)$. The source term is taken non-null in a circular central zone only in order to avoid numerical differentiation problems in the case of a punctual source.

We generate 300 sequences of 64×64 frames of length 150 from this setting by assimilating pixel $(i, j) \in [\![0, 63]\!] \times [\![0, 63]\!]$ to a point $(x, y) \in [0, 63] \times [0, 63]$ and selecting a frame per time interval of size 0.002. This discretization is used to solve Equation (C.19) as its spatial derivatives are estimated thanks to finite differences; once computed, they are used in an ODE numerical solver to solve Equation (C.19) on t. Spatial derivatives are estimated with finite differences of order 5, and the ODE solver is the fourth-order Runge-Kutta method with the $3/8$ rule (Kutta, 1901; Hairer, Nørsett, and Wanner, 1993) and step size 0.001. The data are finally normalized following a min-max $[0, 1]$ scaling per sequence.

The dataset is then split into training (240 sequences) and testing (60 sequences) sets. Sequences sampled during training are random chunks of length $\nu + 1 = 25$, including $\tau + 1 = 5$ conditioning frames, of full-size training sequences. Sequences used during testing are all possible chunks of length $\tau + 1 + 40 = 45$ from full-size testing sequences.

Finally, WaveEq-100 is created from WaveEq by selecting 100 pixels uniformly at random. The extracted pixels are selected before training and are fixed for both training and testing. Therefore, training and testing sequences for WaveEq-100 consist of vectors of size 100 extracted from WaveEq frames. Training and testing sequences are chosen to be the same as those of WaveEq.

C.4.2. Sea Surface Temperature (SST)

SST is composed of sea surface temperatures of the Atlantic ocean generated using E.U. Copernicus Marine Service Information thanks to the state-of-the-art simulation engine NEMO. The use of a so-called reanalysis procedure implies that these data accurately represent the actual temperature measures. For more information, we refer to the complete description of the data by de Bézenac, Pajot, and Gallinari (2018). The data history of this engine is available online at the Copernicus Marine Service.[1] Unfortunately, due to recent maintenance, data history is limited to the last three years; prior histories should be manually requested.

The dataset uses daily temperature acquisitions from Thursday 28$^{\text{th}}$ December, 2006 to Wednesday 5$^{\text{th}}$ April, 2017 of a 481×781 zone, from which 29 zones of size 64×64 zones are extracted. We follow the same setting as de Bézenac, Pajot, and Gallinari (2018) by training all models with $\tau + 1 = 4$ conditioning steps and $\nu - \tau = 6$ steps to predict, and evaluating them only on zones 17 to 20. These zones are particularly interesting since they are the places where cold waters meet warm waters, inducing more pronounced motion.

We normalize the data in the same manner as de Bézenac, Pajot, and Gallinari (2018). Each daily acquisition of a zone is first normalized using the mean and standard deviation of measured temperatures in this zone computed for all days with the same date of the year from the available data (daily history climatological normalization). Each zone is then normalized so that the mean and variance over all acquisitions correspond to those of a standard Gaussian distribution. These normalized data are

[1] It can be found at the following URL: https://resources.marine.copernicus.eu/?option=com_csw&view=details&product_id=GLOBAL_ANALYSIS_FORECAST_PHY_001_024.

finally fed to the model; MSE scores reported in Table 5.1 are computed once the performed normalization of the data and model prediction is reverted to the original temperature measurement space, in order to compute physically meaningful scores.

Training sequences correspond to randomly selected chunks of length $\nu = 10$ in the first daily 2987 acquisitions (corresponding to 80% of total acquisitions), and testing sequences to all possible chunks of length $\nu = 10$ in the remaining 747 acquisitions.

C.4.3. Moving MNIST

We use the same version of the dataset as in Chapter 4. We train all models in the same setting as Denton and Birodkar (2017), with $\tau + 1 = 5$ conditioning frames and $\nu - \tau = 10$ frames to predict. In the case of variational-based models (SVG and SRVP), we use the same training procedure as in the original works by training the networks to reconstruct sequences of 15 frames, like in standard VAE training. We test all models by making them predict either 10 or 95 frames ahead.

To evaluate disentanglement with content swapping, we report PSNR and SSIM metrics between the swapped sequence produced by our model and a ground truth. However, having two digits in the image, there is an ambiguity as to in which order target digits should be swapped in the ground truth. To account for this ambiguity and thanks to the synthetic nature of the dataset, we instead build two ground truth sequences for both possible digit swap permutations, and report the lowest metric between the generated sequence and both ground truths (i.e. we choose the closest ground truth to compare to with respect to the considered metric).

C.4.4. 3D Warehouse Chairs

This multi-view dataset introduced by Aubry et al. (2014) contains 1393 three-dimensional models of chairs seen under the same periodic angles. We resize the original 600×600 images by center-cropping them to 400×400 images, downsample them to 64×64 frames using the Lanczos filter of the Pillow library,[2] and linearly transform their pixel values from the integer range $[\![0, 255]\!]$ to $[0, 1]$.

We create sequences from this dataset for our model by assembling the views of each chair to simulate its rotation from right to left until it reaches its initial position. This process is repeated for each existing angle to serve as initial position for all chairs. We choose this dataset instead of Denton and Birodkar (2017)'s multi-view chairs dataset because the latter contains too few objects to allow both tested methods to generalize on the testing set, preventing us to draw any conclusion from the experiment. We train models on this dataset with $\tau + 1 = 5$ conditioning frames and $\nu - \tau = 10$ frames to predict, and test them to predict 15 frames within the content swap experiment. Training and testing data are constituted by randomly selecting 85% of the chairs for training and 15% of the remaining ones for testing. Disentanglement metrics are computed similarly to the ones on Moving MNIST, but with only one reference ground truth corresponding to the chair given as content input with the orientation of the chair given as dynamic input.

[2] `https://pillow.readthedocs.io/`

C.4.5. TaxiBJ

This crowd flow dataset provided by Junbo Zhang, Zheng, and Qi (2017) consists in two-channel 32×32 frames representing the inflow and outflow of taxis in Beijing, each pixel corresponding to a square region of the city. Observations are registered every thirty minutes. It is highly structured as the flows are dependent on the infrastructure of the city, and complex since methods have to account for non-local dependencies and model subtle changes in the evolution of the flows.

We follow the preprocessing steps of Y. Wang, Z. Gao, et al. (2018) and Le Guen and Thome (2020) by performing a min-max normalization of the data to the $[0, 1]$ range. We train models on this dataset with $\tau + 1 = 4$ conditioning frames and $\nu - \tau = 4$ frames to predict, and test them to predict 4 frames like our competitors on the last four weeks of data which are excluded from the training set. MSE on this dataset is reported in the $[0, 1]$-normalized space and multiplied by a hundred times the dimensionality of a frame, i.e. by $100 \times 32 \times 32 \times 2$.

C.5. Training Details

Along with the public release of our code, we provide in this section sufficient details in order to replicate our results.

C.5.1. Baselines

PKnl. We retrain PKnl (de Bézenac, Pajot, and Gallinari, 2018) on SST using the official implementation and the indicated hyperparameters.

SVG, MIM and DDPAE. We train SVG (Denton and Fergus, 2018), MIM (Y. Wang, Jianjin Zhang, et al., 2019) and DDPAE (Hsieh et al., 2018) on our version of Moving MNIST using the official implementation and the same hyperparameters that the original authors use for the original version of Moving MNIST.

We train MIM on SST using the recommended hyperparameters of the authors, and SVG by retaining the same hyperparameters as those used on KTH.

DrNet. We train DrNet (Denton and Birodkar, 2017) on our version of Moving MNIST using the same hyperparameters originally used for the first version of the dataset (with digits of different colors). To this end, we reimplement the official Lua code into a Python code in order to train it with a more recent infrastructure. We also train DrNet on 3D Warehouse Chairs using the same hyperparameters used by its authors on the smaller multi-view chairs dataset of the original article.

PhyDNet. We train PhyDNet (Le Guen and Thome, 2020) on SST and our version of Moving MNIST using the official implementation and the same hyperparameters that the authors use for SST and the original version of Moving MNIST. We remove the skip connections used by the authors on the Moving MNIST dataset in order to

perform a fair comparison with other models, such as ours, that do not incorporate skip connections on this dataset.

C.5.2. Model Specifications

C.5.2.1. Architecture

Combination of $\boldsymbol{w}$ and $\boldsymbol{y}$. As explained in Section 5.3, the default choice of combination of $\boldsymbol{w}$ and $\boldsymbol{y}$ as decoder inputs is the concatenation of both vectorial variables: it is generic and allows the decoder to learn an appropriate combination function ζ as in Equation (5.7).

Nonetheless, further knowledge of the studied dataset can help to narrow the choices of combination functions. Indeed, we choose to multiply $\boldsymbol{w}$ and $\boldsymbol{y}$ before giving them as input to the decoder for both datasets WaveEq and WaveEq-100, given the knowledge of the existence of product-separable solutions to the homogeneous version of equation (i.e. without source). This shows that it is possible to change the combination function of $\boldsymbol{w}$ and $\boldsymbol{y}$, and that existing combination functions in the PDE literature could be leveraged for other datasets.

Encoders e_θ^S and e_θ^T, and decoder g_θ. For WaveEq, the encoder and decoder outputs are considered to be vectors; images are thus flattened before encoding and reshaped after decoding to 64×64 frames. The encoder is an MLP with two hidden layers of size 1200 and internal ReLU activation functions. The decoder is an MLP with three hidden layers of size 1200, internal ReLU activation functions, and a final sigmoid activation function. The encoder and decoder used for WaveEq-100 are similar to those used for WaveEq, but with two hidden layers each, of respective sizes 2400 and 150.

We use for SST a VGG16 architecture (Simonyan and Zisserman, 2015), mirrored between the encoder and the decoder, complemented with skip connections integrated into S (Ronneberger, Fischer, and Brox, 2015) from all internal layers of the encoder to corresponding decoder layers, also leveraged by de Bézenac, Pajot, and Gallinari (2018) in their PKnl model. We adapt this VGG16 architecture without skip connections for the 32×32 frames of TaxiBJ by removing the shallowest upsampling and downsampling operations in the VGG encoder and decoder. For Moving MNIST, the encoder and its mirrored decoder are shaped with the DCGAN discriminator and generator architecture (Radford, Metz, and Chintala, 2016), with an additional sigmoid activation after the very last layer of the decoder; this encoder and decoder DCGAN architecture is also used by DrNet, DDPAE and SRVP. We highlight that we leverage in both SST and Moving MNIST architectural choices that are also used in compared baselines, enabling fair comparisons.

For the two-dimensional latent space experiments on SST (see Appendix C.6.2), we use a modified version of the VGG encoder / decoder network by removing the two deepest maximum pooling layers, thus preserving the two-dimensional latent structures. The decoder mirrors the encoder complemented with skip connections.

Regarding 3D Warehouse Chairs, we also follow the same architectural choices as DrNet with a ResNet18-like architecture for the encoder, and a DCGAN architecture

followed by a sigmoid activation after the last layer for the decoder.

Encoders e_θ^S and e_θ^T taking as input multiple observations, we combine them by either concatenating them for the vectorial observations of WaveEq-100, or grouping them on the color channel dimensions for the other datasets where observations are frames. Each encoder and decoder layer was initialized from a normal distribution with standard deviation 0.02 (except for biases initialized to 0, and batch normalizations weights drawn from a Gaussian distribution with unit mean and a standard deviation of 0.02).

ODE solver. We use a residual network as an integrator for Equation (5.10). This residual network is composed of a given number K of residual blocks, each block $i \in [\![1, K]\!]$ implementing the application $\mathrm{id} + h_i$, where h_i is an MLP with a two hidden layers of size H and internal ReLU activation functions. The parameter values for each dataset are:

- for WaveEq and WaveEq-100: $K = 3$ and $H = 512$;
- for SST (with linear latent states): $K = 3$ and $H = 1024$;
- for Moving MNIST, 3D Warehouse Chairs and TaxiBJ: $K = 1$ and $H = 512$.

Each MLP is orthogonally initialized with the following gain for each dataset:

- for WaveEq, WaveEq-100, SST (with linear latent states), 3D Warehouse Chairs and TaxiBJ: 0.71;
- for Moving MNIST: 1.41.

For SST with two-dimensional states, the MLPs are replaced with convolutional layers with kernel size 3, padding 1 and a number of hidden channels equal to $H = 128$. We set $K = 2$ and choose an orthogonal initialization gain of 0.2. ReLU activations are replaced with Leaky ReLU activations and preceded by batch normalization layers.

Latent variable sizes. $\boldsymbol{w}$ and $\boldsymbol{y}$ have the following vectorial dimensions for each dataset:

- for WaveEq and WaveEq-100: 32;
- for SST, respectively $196 \times 16 \times 16$ and $64 \times 16 \times 16$; for the linear version, both are set to 256.
- for Moving MNIST and TaxiBJ: respectively, 128 and 20;
- for 3D Warehouse Chairs: respectively, 128 and 10.

Note that, in order to perform fair comparisons, the size of $\boldsymbol{y}$ for baselines without static component $\boldsymbol{w}$ is chosen to be the sum of the vectorial sizes of $\boldsymbol{w}$ and $\boldsymbol{y}$ in the full model. The skip connections of $\boldsymbol{w}$ for SST cannot, however, be integrated into $\boldsymbol{y}$, as its evolution is only modeled in the latent space, and it is out of the scope of this work to leverage low-level dynamics.

C.5.3. Optimization

Optimization is performed using the Adam optimizer (Kingma and Ba, 2015) with initial learning rate 4×10^{-4} for WaveEq, WaveEq-100, Moving MNIST, 3D Warehouse Chairs and SST and 4×10^{-5} for TaxiBJ, and with decay rates $\beta_1 = 0.9$ (except for the experiments on Moving MNIST where we choose $\beta_1 = 0.5$) and $\beta_2 = 0.99$.

Loss function. Chosen coefficients values of λ_{pred}, λ_{AE}, λ^S_{reg}, and λ^T_{reg} are the following:

- $\lambda_{\text{pred}} = 45$;
- $\lambda_{\text{AE}} = 45$ for TaxiBJ; 10 for SST (linear)and Moving MNIST; 1 for WaveEq, WaveEq-100 and 3D Warehouse Chairs; 0.1 for SST;
- $\lambda^S_{\text{reg}} = 100$ for SST; $\lambda^S_{\text{reg}} = 45$ for WaveEq, WaveEq-100, SST (linear) and Moving MNIST; 1 for 3D Warehouse Chairs; 0.0001 for TaxiBJ;
- $\lambda^T_{\text{reg}} = \frac{1}{2}p \times 10^{-3}$ for WaveEq, WaveEq-100, Moving MNIST, 3D Warehouse Chairs and TaxiBJ (where p is the dimension of T); $\frac{1}{2}p \times 10^{-2}$ for SST (linear); 5×10^{-6} for SST.

The batch size is chosen to be 128 for WaveEq, WaveEq-100, Moving MNIST and 3D Warehouse Chairs, and 100 for SST and TaxiBJ.

Training length. The number of training epochs for each dataset is:

- for WaveEq and WaveEq-100: 250 epochs;
- for SST: 30 epochs; for SST (linear): 80 epochs;
- for Moving MNIST: 800 epochs, with an epoch corresponding to 200 000 trajectories (the dataset being infinite), and with the learning rate successively divided by 2 at epochs 300, 400, 500, 600, and 700;
- for 3D Warehouse Chairs: 120 epochs;
- for TaxiBJ: 550 epochs, with the learning rate divided by 5 at epochs 250, 300, 350, 400 and 450.

C.5.4. Prediction Offset for SST

Using the formalism of our work, our algorithm trains to predict $\boldsymbol{x} = (\boldsymbol{x}_{t_0}, \dots, \boldsymbol{x}_{t_1})$ from conditioning frames $\boldsymbol{x}_{t_0::\tau}$. Therefore, it first learns to reconstruct $\boldsymbol{x}_{t_0::\tau}$.

However, the evolution of SST data is chaotic and predicting above a horizon of 6 with coherent and sharp estimations is challenging. Therefore, for the SST dataset only, we chose to supervise the prediction from $t = t_0 + (\tau + 1)\Delta t$, i.e our algorithm trains to forecast $\boldsymbol{x}_{t_0+(\tau+1)\Delta t}, \dots, \boldsymbol{x}_{t_1}$ from $\boldsymbol{x}_{t_0::\tau}$. It simply consists in making the dynamic representation $e^T_\theta(\boldsymbol{x}_{t_0::\tau})$ temporally match the observation $\boldsymbol{x}_{t_0+(\tau+1)\Delta t}$ instead of $\boldsymbol{x}_{t_0}$. This index offset does not change our interpretation of spatiotemporal disentanglement through the separation of variables.

Table C.1.: FVD score of compared models on KTH. The bold score indicates the best performing method.

Ours	PhyDNet	SVG	DrNet	SRVP
330	384	375	383	**222**

C.6. Additional Results and Samples

C.6.1. Preliminary Results on KTH

The application of our method to natural videos is an interesting perspective but would motivate further adaptation of the model (cf. Sections 5.5 and 8.2), in particular regarding the integration of stochastic dynamics. Tackling this issue would require incorporating stochasticity in our model, for example leveraging variational autoencoders like in Chapter 4, or supplementing it with adversarial losses on the image space, for instance like M. Mathieu, Couprie, and LeCun (2016) and A. X. Lee, R. Zhang, et al. (2018). These changes are feasible but are out of the scope of this chapter and we leave them as future work.

Nonetheless, we investigate the realistic video dataset KTH that we also study in Chapter 4. We train our model, SVG, DrNet, PhyDNet and SRVP on this dataset. DrNet and PhyDNet are powerful deterministic approaches, while SVG is a standard stochastic video prediction model; SRVP corresponds to the performant model introduced in Chapter 4. We compare all models in terms of FVD (lower is better) similarly to Chapter 4.

Results are reported in Table C.1. We observe that our model substantially outperforms the considered baselines, even though it expectedly remains outmatched by SRVP since the latter is specifically designed for the forecasting of stochastic videos. These significant results against powerful deterministic baselines, and even the standard stochastic method SVG, confirm our advantage at modeling complex dynamics and support our claim that our model lays the foundations for domain-specific methods, such as a stochastic version for natural videos.

Reproducibility. We use the following training parameters for KTH:

- we follow the same dataset processing and evaluation procedure as Denton and Fergus (2018);
- we train our model on 125 epochs with batch size 100, with an epoch being defined as 100 000 training sequences;
- we set the learning rate to 2×10^{-4} and the same optimizer parameters as for SST;
- $\lambda_{\text{pred}} = 45$, $\lambda_{\text{AE}} = 10 = \lambda^S_{\text{reg}} = 10$, $\lambda^T_{\text{reg}} = p \times 10^{-4}$;

Table C.2.: Forecasting performance on SST of PKnl, PhyDNet and our model with respect to indicated prediction horizons in terms of MSE and SSIM. Bold scores indicate the best performing method.

Models	MSE		SSIM	
	$t+6$	$t+10$	$t+6$	$t+10$
PKnl	1.28	2.03	0.6686	0.5844
PhyDNet	1.27	1.91	0.5782	0.4645
SVG	1.51	2.06	0.6259	0.5595
MIM	0.91	1.45	0.7406	0.6525
Ours	**0.86**	**1.43**	**0.7466**	**0.6577**
Ours (without $\boldsymbol{w}$)	0.95	1.50	0.7204	0.6446
Ours (linear)	1.15	1.80	0.6837	0.5984
Ours (linear, without $\boldsymbol{w}$)	1.46	2.19	0.6200	0.5456

- the size of $\boldsymbol{w}$ and $\boldsymbol{y}$ are respectively 128 and 50;
- the ODE is solved with a flat latent architecture and parameters $K = 1$ and $H = 512$;
- the encoder and decoder architecture is VGG16 with skip connections integrated into $\boldsymbol{y}$ from e^S_θ to g_θ, and with the decoder output being given to a final sigmoid activation.

We reproduced SVG, DrNet and PhyDNet using the recommended hyperparameters of their authors. We train PhyDNet for 125 epochs, like our model, to obtain a fair evaluation despite its low efficiency (six times slower than ours).

C.6.2. Modeling SST with Separation of Variables

We present in Table C.2 results of Table 5.1 for SST, complemented with an alternative version of our model obtained using vectorial representation for $\boldsymbol{w}$ and $\boldsymbol{y}$ and MLPs to compute the derivative of $\boldsymbol{y}$. The latter setting corresponds to a strictly enforced separation of spatial and dynamical variables This alternative version already significantly outperforms powerful methods PhyDNet, PKnl and SVG thanks to this separation, as attested by the corresponding ablation without a static component, and despite its reduced capacity compared to the original model.

However, sea surface temperature exhibits a highly local structure that can be assimilated to a flow in a coarse approximation. For example, there is a transport of large bodies of hot and cold water. Accordingly, performances may be enhanced by considering local dependencies in the dynamics, as also implemented by MIM and PhyDNet. We do so in the original model by considering like the latter methods two-dimensional latent states for the static $\boldsymbol{w}$ and the dynamic $\boldsymbol{y}$, and convolutional networks to model the derivative of $\boldsymbol{y}$.

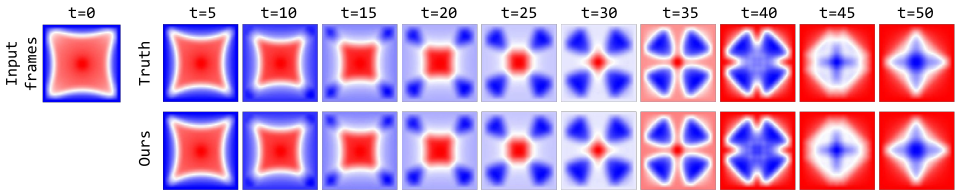


Figure C.1.: Example of predictions of our model on WaveEq.

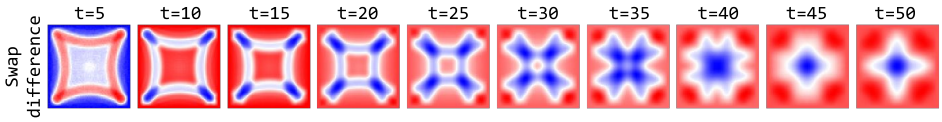


Figure C.2.: Evolution of the scaled difference between the forecast of a sequence and the same forecast with a spatial code coming from another sequence for the WaveEq dataset.

Accounting for such locality in the dynamics amounts to implementing another separation than the usual separation between t and spatial variables. Indeed, it rather excludes unknown content variables from the dynamics. The resulting dynamics is then a PDE over time t as well the observation abscissa and ordinate that we implement using convolutional neural networks, following Z. Long, Y. Lu, X. Ma, et al. (2018) and Ayed et al. (2020). This different kind of separation of variables simplifies learning by estimating a PDE that is simpler than the original one, since it acts on fewer variables. It highlights the generality of our intuition of using the separation of variables, which may be used in other settings that strict spatiotemporal disentanglement. This approach, while still maintaining disentangling properties, significantly improves prediction performances.

Note that our proposition remains computationally much lighter than the alternatives MIM, PhyDNet and SVG.

C.6.3. Additional Samples

C.6.3.1. WaveEq

We provide in Figure C.1 a sample for the WaveEq dataset, highlighting the long-term consistency in the forecasts of our algorithm.

We also show in Figure C.2 the effect on forecasting of changing the spatial code $\boldsymbol{w}$ with the one of another sequence.

C.6.3.2. SST

We provide an additional sample for SST in Figure C.3.

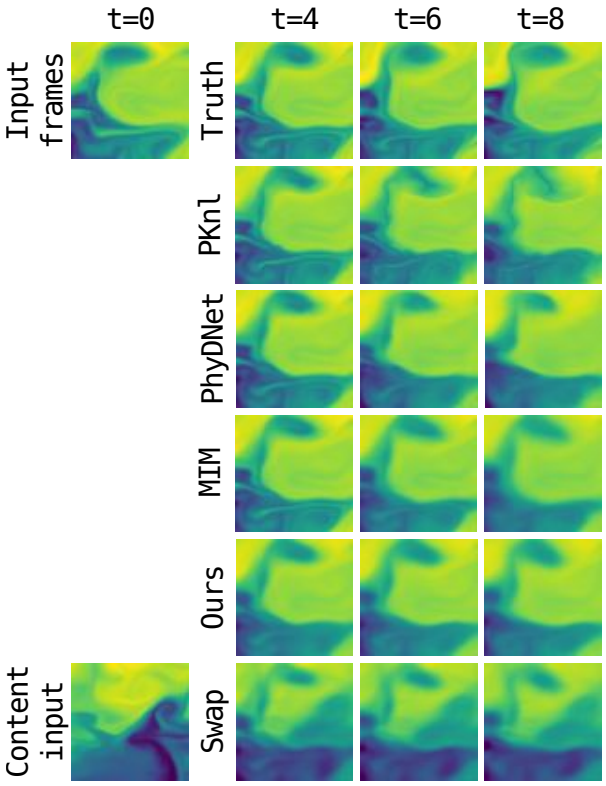


Figure C.3.: Example of predictions of compared models on SST.

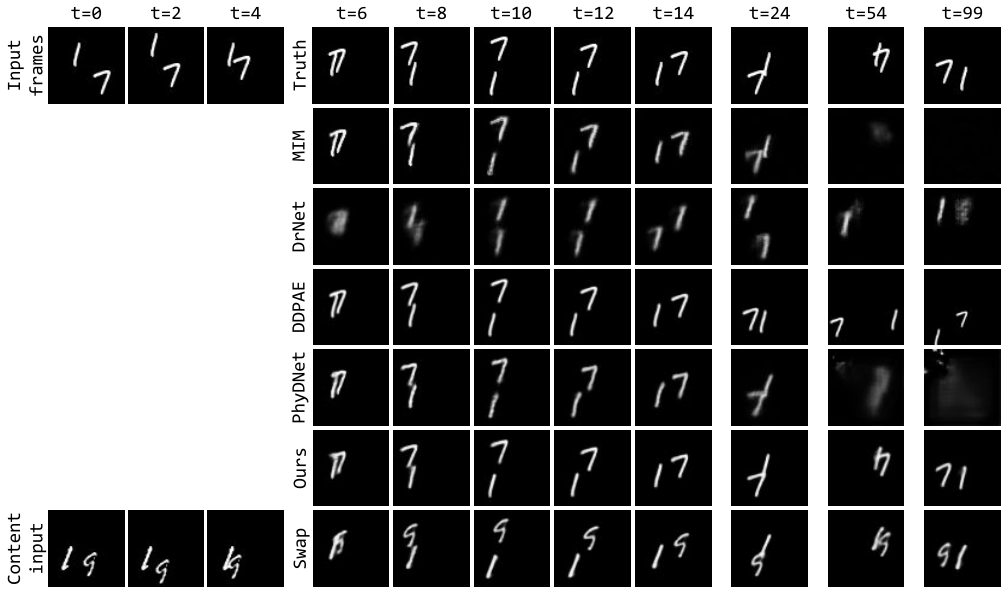


Figure C.4.: Example of predictions of compared models on Moving MNIST.

C.6.3.3. Moving MNIST

We provide two additional samples for Moving MNIST in Figures C.4 and C.5.

C.6.3.4. 3D Warehouse Chairs

We provide a qualitative comparison for the content swap experiment between our model and DrNet for 3D Warehouse Chairs in Figure C.6. We notice that DrNet produces substantially more blurry samples than our model and has difficulties capturing the exact dynamics of the chairs.

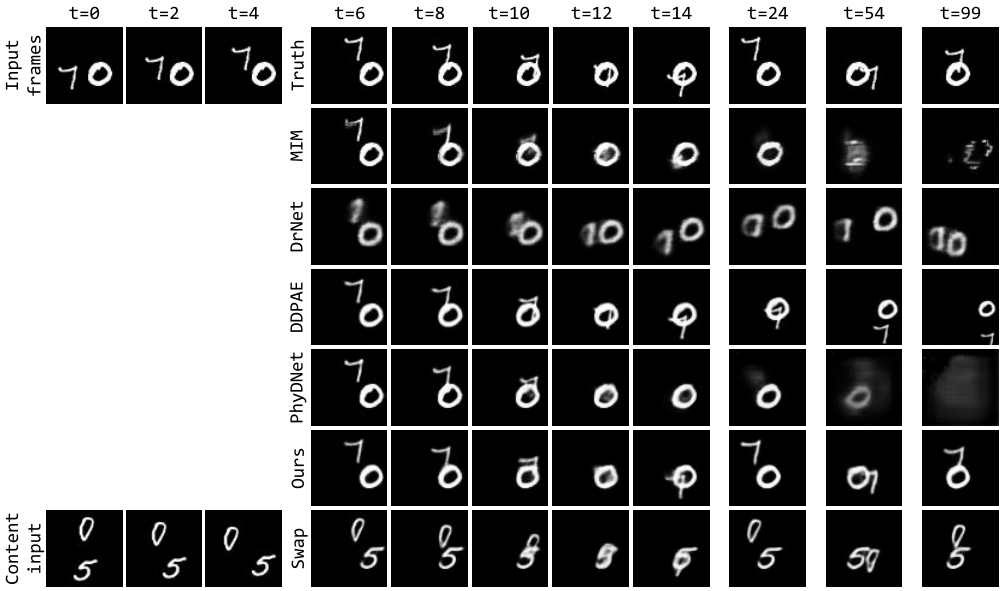
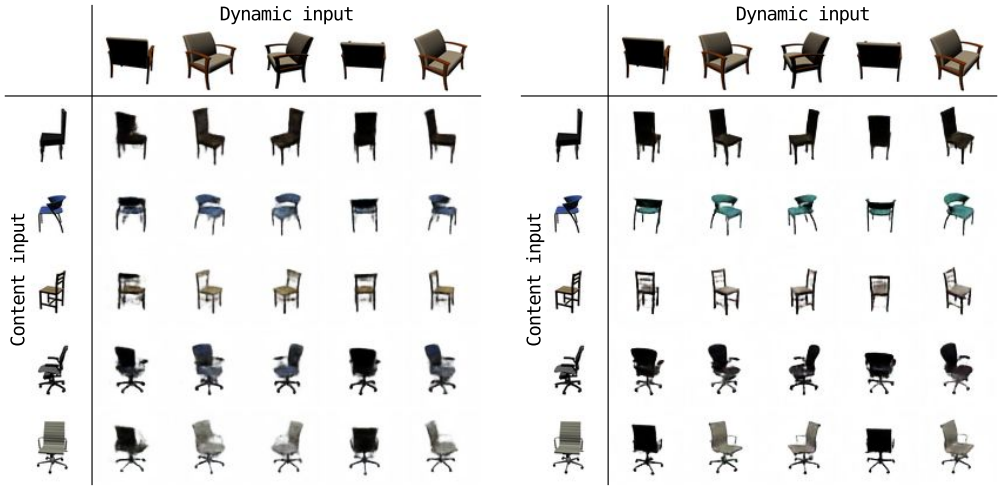


Figure C.5.: Example of predictions of compared models on Moving MNIST.



(a) DrNet. (b) Ours.

Figure C.6.: Fusion of content (first column) and dynamic (first row) variables in DrNet and our model on 3D Warehouse Chairs.

Appendix D.

Supplementary Material of Chapter 6

In this appendix chapter of Chapter 6, we provide complete proofs (in Appendix D.1), discussions (in Appendix D.2) and experimental details (in Appendix D.3) for the material presented in Chapter 6.

In the course of this chapter, we drop the subscript g for $\hat{\gamma}_g$, $\hat{\alpha}_g$ and other notations when the dependency on a fixed generator g is clear and indicated in the main paper, for the sake of clarity.

D.1. Proofs of Theoretical Results and Additional Results

We prove in this section all theoretical results mentioned in Sections 6.4 and 6.5. Appendix D.1.2 is devoted to the proof of Theorem 1, Appendix D.1.3 focuses on proving the differentiability results skimmed in Section 6.4.3, Appendix D.1.4 contains the demonstration of Proposition 3, and Appendices D.1.5 and D.1.6 develop the results presented in Section 6.5.

We will need in the course of these proofs the following standard definition. For any measurable function T and measure μ, $T_\sharp\mu$ denotes the push-forward measure which is defined as $T_\sharp\mu(B) = \mu\big(T^{-1}(B)\big)$, for any measurable set B.

D.1.1. Recall of Assumptions

Assumption 1 (Finite training set)**.** $\hat{\gamma} \in \mathcal{P}(\Omega)$ is a finite mixture of Diracs.

Assumption 2 (Kernel)**.** $k\colon \Omega^2 \to \mathbb{R}$ is a symmetric positive semi-definite kernel with $k \in L^2\big(\Omega^2\big)$.

Assumption 3 (Loss regularity)**.** a and b from Equation (6.2) are differentiable with Lipschitz derivatives over $\mathbb{R}$.

Assumption 4 (Discriminator architecture)**.** The discriminator is a standard architecture (fully connected, convolutional or residual). Any activation ϕ in the network satisfies the following properties:

- ϕ is smooth everywhere except on a finite set D;

- for all $j \in \mathbb{N}$, there exist scalars $\lambda_1^{(j)}$ and $\lambda_2^{(j)}$ such that:

$$\forall x \in \mathbb{R} \setminus D, \left|\phi^{(j)}(x)\right| \leq \lambda_1^{(j)}|x| + \lambda_2^{(j)}, \tag{D.1}$$

  where $\phi^{(j)}$ is the j-th derivative of ϕ.

Assumption 5 (Discriminator regularity)**.** $D = \emptyset$, i.e. ϕ is smooth.

Assumption 6 (Discriminator bias)**.** Linear layers have non-null bias terms. Moreover, for all $x, y \in \mathbb{R}$ such that $x \neq y$, the following holds:

$$\mathbb{E}_{\varepsilon\sim\mathcal{N}(0,1)}\phi(x\varepsilon)^2 \neq \mathbb{E}_{\varepsilon\sim\mathcal{N}(0,1)}\phi(y\varepsilon)^2. \tag{D.2}$$

Remark 4 (Typical activations)**.** Assumptions 4 to 6 cover multiple standard activation functions, including tanh, softplus, ReLU, leaky ReLU and sigmoid.

D.1.2. On the Solutions of Equation (6.9)

The methods used in this section are adaptations to our setting of standard methods of proof. In particular, they can be easily adapted to slightly different contexts, the main ingredient being the structure of the kernel integral operator. Moreover, it is also worth noting that, although we relied on Assumption 1 for $\hat{\gamma}$, the results are essentially unchanged if we take a compactly supported measure γ instead.

We decompose the proof into several intermediate results. Theorem 3 and Proposition 6, stated and demonstrated in this section, correspond when combined to Theorem 1.

Let us first prove the following two intermediate lemmas.

Lemma 1. *Let $\delta T > 0$ and $\mathcal{F}_{\delta T} = \mathcal{C}\Big([0, \delta T], B_{L^2(\hat{\gamma})}(f_0, 1)\Big)$ endowed with the norm:*

$$\forall u \in \mathcal{F}_{\delta T},\ \|u\| = \sup_{t\in[0,\delta T]} \|u_t\|_{L^2(\hat{\gamma})}. \tag{D.3}$$

Then $\mathcal{F}_{\delta T}$ is complete.

Proof. Let $(u^n)_n$ be a Cauchy sequence in $\mathcal{F}_{\delta T}$. For a fixed $t \in [0, \delta T]$:

$$\forall n, m, \|u_t^n - u_t^m\|_{L^2(\hat{\gamma})} \leq \|u^n - u^m\|, \tag{D.4}$$

which shows that $(u_t^n)_n$ is a Cauchy sequence in $L^2(\hat{\gamma})$. $L^2(\hat{\gamma})$ being complete, $(u_t^n)_n$ converges to a $u_t^\infty \in L^2(\hat{\gamma})$. Moreover, for $\varepsilon > 0$, because (u^n) is Cauchy, we can choose N such that:

$$\forall n, m \geq N, \|u^n - u^m\| \leq \varepsilon. \tag{D.5}$$

We thus have that:

$$\forall t, \forall n, m \geq N, \|u_t^n - u_t^m\|_{L^2(\hat{\gamma})} \leq \varepsilon. \tag{D.6}$$

Then, by taking m to ∞, by continuity of the $L^2(\hat{\gamma})$ norm:

$$\forall t, \forall n \geq N, \|u_t^n - u_t^\infty\|_{L^2(\hat{\gamma})} \leq \varepsilon, \tag{D.7}$$

which means that:

$$\forall n \geq N, \|u^n - u^\infty\| \leq \varepsilon. \tag{D.8}$$

so that $(u^n)_n$ tends to u^∞.

Moreover, as:

$$\forall n, \|u_t^n\|_{L^2(\hat{\gamma})} \leq 1, \tag{D.9}$$

we have that $\|u_t^\infty\|_{L^2(\hat{\gamma})} \leq 1$.

Finally, let us consider $s, t \in [0, \delta T]$. We have that:

$$\forall n, \|u_t^\infty - u_s^\infty\|_{L^2(\hat{\gamma})} \leq \|u_t^\infty - u_t^n\|_{L^2(\hat{\gamma})} + \|u_t^n - u_s^n\|_{L^2(\hat{\gamma})} + \|u_s^\infty - u_s^n\|_{L^2(\hat{\gamma})}. \tag{D.10}$$

The first and the third terms can then be taken as small as needed by definition of u^∞ by taking n high enough, while the second can be made to tend to 0 as t tends to s by continuity of u^n. This proves the continuity of u^∞ and shows that $u^\infty \in \mathcal{F}_{\delta T}$. □

Lemma 2. *For any $F \in L^2(\hat{\gamma})$, we have that $F \in L^2(\hat{\alpha})$ and $F \in L^2\left(\hat{\beta}\right)$ with:*

$$\|F\|_{L^2(\hat{\alpha})} \leq \sqrt{2}\|F\|_{L^2(\hat{\gamma})} \text{ and } \|F\|_{L^2\left(\hat{\beta}\right)} \leq \sqrt{2}\|F\|_{L^2(\hat{\gamma})}. \tag{D.11}$$

Proof. For any $F \in L^2(\hat{\gamma})$, we have that

$$\|F\|^2_{L^2(\hat{\gamma})} = \frac{1}{2}\|F\|^2_{L^2(\hat{\alpha})} + \frac{1}{2}\|F\|^2_{L^2\left(\hat{\beta}\right)}, \tag{D.12}$$

so that $F \in L^2(\hat{\alpha})$ and $F \in L^2\left(\hat{\beta}\right)$ with:

$$\|F\|^2_{L^2(\hat{\alpha})} = 2\|F\|^2_{L^2(\hat{\gamma})} - \|F\|^2_{L^2\left(\hat{\beta}\right)} \leq 2\|F\|^2_{L^2(\hat{\gamma})} \tag{D.13}$$

$$\text{and } \|F\|^2_{L^2\left(\hat{\beta}\right)} = 2\|F\|_{L^2(\hat{\gamma})} - \|F\|_{L^2(\hat{\alpha})} \leq 2\|F\|^2_{L^2(\hat{\gamma})}, \tag{D.14}$$

which allows us to conclude. □

From this, we can prove the existence and uniqueness of the initial value problem from Equation (6.9).

Theorem 3 (Existence and Uniqueness)**.** *Under Assumptions 1 to 3, Equation (6.9) with initial value f_0 admits a unique solution $f_\cdot : \mathbb{R}_+ \to L^2(\Omega)$.*

Proof.

A few inequalities. We start this proof by proving a few inequalities.

Let $f, g \in L^2(\hat{\gamma})$. We have, by the Cauchy-Schwarz inequality, for all $z \in \Omega$:

$$\begin{aligned}\left|\left(\mathcal{T}_{k,\hat{\gamma}}\left(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f)\right) - \mathcal{T}_{k,\hat{\gamma}}\left(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(g)\right)\right)(\boldsymbol{z})\right| \\ \leq \left\|k(\boldsymbol{z},\cdot)\right\|_{L^2(\hat{\gamma})}\left\|\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f) - \nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(g)\right\|_{L^2(\hat{\gamma})}.\end{aligned} \tag{D.15}$$

Moreover, by definition:

$$\left\langle \nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f) - \nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(g), h\right\rangle_{L^2(\hat{\gamma})} = \int \left(a'_f - a'_g\right) h \,\mathrm{d}\hat{\alpha} - \int \left(b'_f - b'_g\right) h \,\mathrm{d}\hat{\beta}, \tag{D.16}$$

so that:

$$\begin{aligned}&\left\|\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f) - \nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(g)\right\|^2_{L^2(\hat{\gamma})} \\ &\quad\leq \left\|\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f) - \nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(g)\right\|_{L^2(\hat{\gamma})}\left(\left\|a'_f - a'_g\right\|_{L^2(\hat{\alpha})} + \left\|b'_f - b'_g\right\|_{L^2\left(\hat{\beta}\right)}\right),\end{aligned} \tag{D.17}$$

and then, along with Lemma 2:

$$\begin{aligned}\left\|\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f) - \nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(g)\right\|_{L^2(\hat{\gamma})} &\leq \left\|a'_f - a'_g\right\|_{L^2(\hat{\alpha})} + \left\|b'_f - b'_g\right\|_{L^2\left(\hat{\beta}\right)} \\ &\leq \sqrt{2}\left(\left\|a'_f - a'_g\right\|_{L^2(\hat{\gamma})} + \left\|b'_f - b'_g\right\|_{L^2(\hat{\gamma})}\right).\end{aligned} \tag{D.18}$$

By Assumption 3, we know that a' and b' are Lipschitz with constants that we denote K_1 and K_2. We can then write:

$$\forall \boldsymbol{x},\ \left|a'\big(f(\boldsymbol{x})\big) - a'\big(g(\boldsymbol{x})\big)\right| \leq K_1\left|f(\boldsymbol{x}) - g(\boldsymbol{x})\right| \tag{D.19}$$

$$\text{and } \forall \boldsymbol{x},\ \left|b'\big(f(\boldsymbol{x})\big) - b'\big(g(\boldsymbol{x})\big)\right| \leq K_2\left|f(\boldsymbol{x}) - g(\boldsymbol{x})\right|, \tag{D.20}$$

so that:

$$\left\|a'_f - a'_g\right\|_{L^2(\hat{\gamma})} \leq K_1\|f - g\|_{L^2(\hat{\gamma})}, \qquad \left\|b'_f - b'_g\right\|_{L^2(\hat{\gamma})} \leq K_2\|f - g\|_{L^2(\hat{\gamma})}. \tag{D.21}$$

Finally, we can now write, for all $z \in \Omega$:

$$\left|\left(\mathcal{T}_{k,\hat{\gamma}}\left(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f)\right) - \mathcal{T}_{k,\hat{\gamma}}\left(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(g)\right)\right)(\boldsymbol{z})\right| \leq \sqrt{2}(K_1 + K_2)\|f - g\|_{L^2(\hat{\gamma})}\left\|k(\boldsymbol{z},\cdot)\right\|_{L^2(\hat{\gamma})}, \tag{D.22}$$

and then:

$$\left\|\mathcal{T}_{k,\hat{\gamma}}\left(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f)\right) - \mathcal{T}_{k,\hat{\gamma}}\left(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(g)\right)\right\|_{L^2(\hat{\gamma})} \leq K\|f - g\|_{L^2(\hat{\gamma})}, \tag{D.23}$$

where

$$K = \sqrt{2}(K_1 + K_2)\sqrt{\int \left\|k(\boldsymbol{z}, \cdot)\right\|^2_{L^2(\hat{\gamma})} \mathrm{d}\hat{\gamma}(\boldsymbol{z})} \tag{D.24}$$

is finite as a finite sum of finite terms from Assumptions 1 and 2. In particular, putting $g = 0$ and using the triangular inequality also gives us:

$$\left\|\mathcal{T}_{k,\hat{\gamma}}\left(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f)\right)\right\|_{L^2(\hat{\gamma})} \leq K\|f\|_{L^2(\hat{\gamma})} + M, \tag{D.25}$$

where $M = \left\|\mathcal{T}_{k,\hat{\gamma}}\left(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(0)\right)\right\|_{L^2(\hat{\gamma})}$.

Existence and uniqueness in $L^2(\hat{\gamma})$. We now adapt the standard fixed point proof to prove the existence and uniqueness of a solution to the studied equation in $L^2(\hat{\gamma})$.

We consider the family of spaces $\mathcal{F}_{\delta T} = \mathcal{C}\Big([0, \delta T], B_{L^2(\hat{\gamma})}(f_0, 1)\Big)$. $\mathcal{F}_{\delta T}$ is defined, for $\delta T > 0$, as the space of continuous functions from $[0, \delta T]$ to the closed ball of radius 1 centered around f_0 in $L^2(\hat{\gamma})$ which we endow with the norm:

$$\forall u \in \mathcal{F}_{\delta T},\ \|u\| = \sup_{t\in[0,\delta T]} \|u_t\|_{L^2(\hat{\gamma})}. \tag{D.26}$$

We now define the application Φ where $\Phi(u)$ is defined as, for any $u \in \mathcal{F}_{\delta T}$:

$$\Phi(u)_t = f_0 + \int_0^t \mathcal{T}_{k,\hat{\gamma}}\left(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(u_s)\right) \mathrm{d}s. \tag{D.27}$$

We have, using Equation (D.25):

$$\left\|\Phi(u)_t - f_0\right\|_{L^2(\hat{\gamma})} \leq \int_0^t \Big(K\|u_s\|_{L^2(\hat{\gamma})} + M\Big)\, \mathrm{d}s \leq (K + M)\delta T. \tag{D.28}$$

Thus, taking $\delta T = \big(2(K + M)\big)^{-1}$ makes Φ an application from $\mathcal{F}_{\delta T}$ into itself. Moreover, we have:

$$\forall u, v \in \mathcal{F}_{\delta T}, \left\|\Phi(u) - \Phi(v)\right\| \leq \frac{1}{2}\|u - v\|, \tag{D.29}$$

which means that Φ is a contraction of $\mathcal{F}_{\delta T}$. Lemma 1 and the Banach-Picard theorem (Banach, 1922) then tell us that Φ has a unique fixed point in $\mathcal{F}_{\delta T}$. It is then obvious that such a fixed point is a solution of Equation (6.9) over $[0, \delta T]$.

Let us now consider the maximal $T > 0$ such that a solution f_t of Equation (6.9) is defined over $[0, T)$. We have, using Equation (D.25):

$$\forall t \in [0, T), \|f_t\|_{L^2(\hat{\gamma})} \leq \|f_0\|_{L^2(\hat{\gamma})} + \int_0^t \Big(\|f_s\|_{L^2(\hat{\gamma})} + M\Big)\, \mathrm{d}s, \tag{D.30}$$

which, using Grönwall's lemma (Grönwall, 1919), gives:

$$\forall t \in [0, T), \|f_t\|_{L^2(\hat{\gamma})} \leq \|f_0\|_{L^2(\hat{\gamma})} \mathrm{e}^{KT} + \frac{M}{K}\Big(\mathrm{e}^{KT} - 1\Big). \tag{D.31}$$

Define $g^n = f_{T-\frac{1}{n}}$. We have, again using Equation (D.25):

$$\begin{aligned}\forall m \geq n, \|g^n - g^m\|_{L^2(\hat{\gamma})} &\leq \int_{T-\frac{1}{n}}^{T-\frac{1}{m}} (K\|f_s\| + M)\,\mathrm{d}s \\ &\leq \left(\frac{1}{n} - \frac{1}{m}\right)\left(\|f_0\|_{L^2(\hat{\gamma})}\mathrm{e}^{KT} + \frac{M}{K}\left(\mathrm{e}^{KT} - 1\right)\right),\end{aligned} \tag{D.32}$$

which shows that $(g^n)_n$ is a Cauchy sequence. $L^2(\hat{\gamma})$ being complete, we can thus consider its limit g^∞. Clearly, f_t tends to g^∞ in $L^2(\hat{\gamma})$. By considering the initial value problem associated with Equation (6.9) starting from g^∞, we can thus extend the solution f_t to $[0, T + \delta T)$, thus contradicting the maximality of T, which proves that the solution can be extended to $\mathbb{R}_+$.

Existence and uniqueness in $L^2(\Omega)$. We now conclude the proof by extending the previous solution to $L^2(\Omega)$. We keep the same notations as above and, in particular, f is the unique solution of Equation (6.9) with initial value f_0.

Let us define $\tilde{f}$ as:

$$\forall t, \forall \boldsymbol{x}, \tilde{f}_t(\boldsymbol{x}) = f_0(\boldsymbol{x}) + \int_0^t \mathcal{T}_{k,\hat{\gamma}}\left(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f_s)\right)(\boldsymbol{x})\,\mathrm{d}s, \tag{D.33}$$

where the right-hand side only depends on f and is thus well-defined. By remarking that $\tilde{f}$ is equal to f on $\operatorname{supp}\hat{\gamma}$ and that, for every s,

$$\mathcal{T}_{k,\hat{\gamma}}\left(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}\left(\tilde{f}_s\right)\right) = \mathcal{T}_{k,\hat{\gamma}}\left(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}\left(\tilde{f}_s\Big|_{\operatorname{supp}\hat{\gamma}}\right)\right) = \mathcal{T}_{k,\hat{\gamma}}\left(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f_s)\right), \tag{D.34}$$

we see that $\tilde{f}$ is solution to Equation (6.9). Moreover, from Assumption 2, we know that, for any $\boldsymbol{z} \in \Omega$, $\int k(\boldsymbol{z}, \boldsymbol{x})^2\,\mathrm{d}\Omega(x)$ is finite and, from Assumption 1, that $\left\|k(\boldsymbol{z}, \cdot)\right\|^2_{L^2(\hat{\gamma})}$ is a finite sum of terms $k(\boldsymbol{z}, \boldsymbol{x}_i)^2$ which shows that $\int \left\|k(\boldsymbol{z}, \cdot)\right\|^2_{L^2(\hat{\gamma})}\,\mathrm{d}\Omega(\boldsymbol{z})$ is finite, again from Assumption 2. We can then say that $\tilde{f}_s \in L^2(\Omega)$ for any s by using the above with Equation (D.22) taken for $g = 0$.

Finally, suppose h is a solution to Equation (6.9) with initial value f_0. We know that $h|_{\operatorname{supp}\hat{\gamma}}$ coincides with f and thus with $\tilde{f}\Big|_{\operatorname{supp}\hat{\gamma}}$ in $L^2(\hat{\gamma})$ as we already proved uniqueness in the latter space. Thus, we have that $\left\| h_s|_{\operatorname{supp}\hat{\gamma}} - \tilde{f}_s\Big|_{\operatorname{supp}\hat{\gamma}} \right\|_{L^2(\hat{\gamma})} = 0$ for any s. Now, we have:

$$\begin{aligned}\forall \boldsymbol{z} \in \Omega, \forall s, &\left|\left(\mathcal{T}_{k,\hat{\gamma}}\left(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(h_s)\right) - \mathcal{T}_{k,\hat{\gamma}}\left(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}\left(\tilde{f}_s\right)\right)\right)(\boldsymbol{z})\right| \\ &= \left|\left(\mathcal{T}_{k,\hat{\gamma}}\left(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}\left(h_s|_{\operatorname{supp}\hat{\gamma}}\right)\right) - \mathcal{T}_{k,\hat{\gamma}}\left(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}\left(\tilde{f}_s\Big|_{\operatorname{supp}\hat{\gamma}}\right)\right)\right)(\boldsymbol{z})\right| \leq 0,\end{aligned} \tag{D.35}$$

by Equation (D.22). This shows that $\partial_t\Big(\tilde{f} - h\Big) = 0$ and, given that $h_0 = \tilde{f}_0 = f_0$, we have $h = \tilde{f}$ which concludes the proof.

□

There only remains to prove for Theorem 1 the inversion between the integral over time and the integral operator. We first prove an intermediate lemma and then conclude with the proof of the inversion.

Lemma 3. *Under Assumptions 1 to 3,*

$$\int_0^T \left(\left\| a' \right\|_{L^2\left((f_s)_\sharp \hat{\alpha}\right)} + \left\| b' \right\|_{L^2\left((f_s)_\sharp \hat{\beta}\right)} \right) \mathrm{d}s \tag{D.36}$$

is finite for any $T > 0$.

Proof. Let $T > 0$. We have, by Assumption 3 and the triangular inequality:

$$\forall \boldsymbol{x}, \left| a'\big(f(\boldsymbol{x})\big) \right| \leq K_1 \left| f(\boldsymbol{x}) \right| + M_1, \tag{D.37}$$

where $M_1 = \left| a'(0) \right|$. We can then write, using Lemma 2 and the inequality from Equation (D.31):

$$\begin{aligned} \forall s \leq T, \left\| a' \right\|_{L^2\left((f_s)_\sharp \hat{\alpha}\right)} &\leq K_1 \sqrt{2} \| f_s \|_{L^2(\hat{\gamma})} + M_1 \\ &\leq K_1 \sqrt{2} \left(\| f_0 \|_{L^2(\hat{\gamma})} \mathrm{e}^{KT} + \frac{M}{K} \Big(\mathrm{e}^{KT} - 1 \Big) \right) + M_1, \end{aligned} \tag{D.38}$$

the latter being constant with respect to s and thus integrable on $[0, T]$. We can then bound $\left\| b' \right\|_{L^2\left((f_s)_\sharp \hat{\beta}\right)}$ similarly, which concludes the proof. □

Proposition 6 (Integral inversion)**.** *Under Assumptions 1 to 3, the following integral inversion holds:*

$$f_t = f_0 + \int_0^t \mathcal{T}_{k_f, \hat{\gamma}} \Big(\nabla^{\hat{\gamma}} \mathcal{L}_{\hat{\alpha}, \hat{\beta}}(f_s) \Big) \, \mathrm{d}s = f_0 + \mathcal{T}_{k_f, \hat{\gamma}} \left(\int_0^t \nabla^{\hat{\gamma}} \mathcal{L}_{\hat{\alpha}, \hat{\beta}}(f_s) \, \mathrm{d}s \right). \tag{D.39}$$

Proof. By definition, a straightforward computation gives, for any function $h \in L^2(\hat{\gamma})$:

$$\Big\langle \nabla^{\hat{\gamma}} \mathcal{L}_{\hat{\alpha}}(f), h \Big\rangle_{L^2(\hat{\gamma})} = \mathrm{d}\mathcal{L}_{\hat{\alpha}}(f)[h] = \int a'_f h \, \mathrm{d}\hat{\alpha} - \int b'_f h \, \mathrm{d}\hat{\beta}. \tag{D.40}$$

We can then write:

$$\begin{aligned} \left\| \nabla^{\hat{\gamma}} \mathcal{L}_{\hat{\alpha}}(f_t) \right\|^2_{L^2(\hat{\gamma})} &= \Big\langle \nabla^{\hat{\gamma}} \mathcal{L}_{\hat{\alpha}}(f_t), \nabla^{\hat{\gamma}} \mathcal{L}_{\hat{\alpha}}(f_t) \Big\rangle_{L^2(\hat{\gamma})} \\ &= \int a'_{f_t} \nabla^{\hat{\gamma}} \mathcal{L}_{\hat{\alpha}}(f_t) \, \mathrm{d}\hat{\alpha} - \int b'_{f_t} \nabla^{\hat{\gamma}} \mathcal{L}_{\hat{\alpha}}(f_t) \, \mathrm{d}\hat{\beta}, \end{aligned} \tag{D.41}$$

so that, with the Cauchy-Schwarz inequality and Lemma 2:

$$\begin{aligned}\left\|\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f_t)\right\|^2_{L^2(\hat{\gamma})} &\leq \int \left|a'_{f_t}\right|\left|\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f_t)\right| \mathrm{d}\hat{\alpha} + \int \left|b'_{f_t}\right|\left|\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f_t)\right| \mathrm{d}\hat{\beta} \\ &\leq \left\|a'_{f_t}\right\|_{L^2(\hat{\alpha})}\left\|\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f_t)\right\|_{L^2(\hat{\alpha})} + \left\|b'_{f_t}\right\|_{L^2\left(\hat{\beta}\right)}\left\|\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f_t)\right\|_{L^2\left(\hat{\beta}\right)} \\ &\leq \sqrt{2}\left\|\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f_t)\right\|_{L^2(\hat{\gamma})}\left[\left\|a'_{f_t}\right\|_{L^2(\hat{\alpha})} + \left\|b'_{f_t}\right\|_{L^2\left(\hat{\beta}\right)}\right],\end{aligned} \tag{D.42}$$

which then gives us:

$$\left\|\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f_t)\right\|_{L^2(\hat{\gamma})} \leq \sqrt{2}\left[\left\|a'\right\|_{L^2\left((f_t)_\sharp\hat{\alpha}\right)} + \left\|b'\right\|_{L^2\left((f_t)_\sharp\hat{\beta}\right)}\right]. \tag{D.43}$$

By the Cauchy-Schwarz inequality and Equation (D.43), we then have for all z:

$$\begin{aligned}\int_0^t \int_{\boldsymbol{x}} \left|k(\boldsymbol{z},\boldsymbol{x})\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f_s)(\boldsymbol{x})\right| \mathrm{d}\hat{\gamma}(\boldsymbol{x})\,\mathrm{d}s &\leq \int_0^t \left\|k(\boldsymbol{z},\cdot)\right\|_{L^2(\hat{\gamma})}\left\|\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f_s)\right\|_{L^2(\hat{\gamma})}\mathrm{d}s \\ &\leq \sqrt{2}\left\|k(\boldsymbol{z},\cdot)\right\|_{L^2(\hat{\gamma})}\int_0^t \left[\left\|a'\right\|_{L^2\left((f_s)_\sharp\hat{\alpha}\right)} + \left\|b'\right\|_{L^2\left((f_s)_\sharp\hat{\beta}\right)}\right]\mathrm{d}s.\end{aligned} \tag{D.44}$$

The latter being finite by Lemma 3, we can now use Fubini's theorem (Fubini, 1907) to conclude that:

$$\begin{aligned}\int_0^t \mathcal{T}_{k_f,\hat{\gamma}}\left(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f_s)\right)\mathrm{d}s &= \int_0^t \int_{\boldsymbol{x}} k(\cdot,\boldsymbol{x})\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f_s)(\boldsymbol{x})\,\mathrm{d}\hat{\gamma}(\boldsymbol{x})\,\mathrm{d}s \\ &= \int_{\boldsymbol{x}} k(\cdot,\boldsymbol{x})\left[\int_0^t \nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f_s)(\boldsymbol{x})\,\mathrm{d}s\right]\mathrm{d}\hat{\gamma}(\boldsymbol{x}) \\ &= \mathcal{T}_{k_f,\hat{\gamma}}\left(\int_0^t \nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f_s)(\boldsymbol{x})\,\mathrm{d}s\right).\end{aligned} \tag{D.45}$$

□

D.1.3. Differentiability of Infinite-Width Networks and their NTKs

Given Theorem 1, establishing the desired differentiability of f_t can be done by separately proving similar results on both $f_t - f_0$ and f_0.

In both cases, this involves the differentiability of the following activation kernel $\mathcal{K}_\phi(A)$ given another differentiable kernel A:

$$\mathcal{K}_\phi(A)\colon \boldsymbol{x},\boldsymbol{y} \mapsto \mathbb{E}_{f\sim\mathcal{GP}(0,A)}\Big[\phi\big(f(\boldsymbol{x})\big)\phi\big(f(\boldsymbol{y})\big)\Big], \tag{D.46}$$

where $\mathcal{GP}(0,A)$ is a univariate centered Gaussian Process (GP) with covariance function A. Indeed, the kernel-transforming operator $\mathcal{K}_\phi$ is central in the recursive computation

of the neural network conjugate kernel sss which determines the NTK (involved in $f_t - f_0 \in \mathcal{H}_k^{\hat{\gamma}_g}$) as well as the behavior of the network at initialization (which follows a GP with the conjugate kernel as covariance).

Hence, our proof of Theorem 2 relies on the preservation of kernel smoothness through $\mathcal{K}_\phi$, proved in Appendix D.1.3.1, which ensures the smoothness of the conjugate kernel, the NTK and, in turn, of f_t as addressed in Appendix D.1.3.2 which concludes the overall proof.

Before developing these two main steps, we first need to state the following lemma showing the regularity of samples of a GP from the regularity of the corresponding kernel.

Lemma 4 (GP regularity)**.** *Let $A: \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be a symmetric kernel. Let V an open set such that A is $\mathcal{C}^\infty$ on $V \times V$. Then the GP induced by the kernel A has a.s. $\mathcal{C}^\infty$ sample paths on V.*

Proof. Because A is $\mathcal{C}^\infty$ on $V \times V$, we know, from Theorem 2.2.2 of Adler (1981) for example, that the corresponding GP f is mean-square smooth on V. If we take α a k-th order multi-index, we also know, again from Adler (1981), that $\partial^\alpha f$ is also a GP with covariance kernel $\partial^\alpha A$. As A is $\mathcal{C}^\infty$, $\partial^\alpha A$ then is differentiable and $\partial^\alpha f$ has partial derivatives which are mean-square continuous. Then, by the Corollary 5.3.12 of Scheuerer (2009), we can say that $\partial^\alpha f$ has continuous sample paths a.s. which means that $f \in \mathcal{C}^k(V)$. This proves the lemma. □

D.1.3.1. $\mathcal{K}_\phi$ Preserves Kernel Differentiability

Given the definition of $\mathcal{K}_\phi(A)$ in Equation (D.46), we choose to prove its differentiability via the dominated convergence theorem and Leibniz integral rule. This requires to derive separate proofs depending on whether ϕ is smooth everywhere or almost everywhere.

The former case allows us to apply strong GP regularity results leading to $\mathcal{K}_\phi$ preserving kernel smoothness without additional hypothesis in Lemma 5. The latter case requires a careful decomposition of the expectation of Equation (D.46) via two-dimensional Gaussian sampling to circumvent the non-differentiability points of ϕ, yielding additional constraints on kernels A for $\mathcal{K}_\phi$ to preserve their smoothness in Lemma 6; these constraints are typically verified in the case of neural networks with bias (cf. Appendix D.1.3.2).

In any case, we emphasize that these differentiability constraints may not be tight and are only sufficient conditions ensuring the smoothness of $\mathcal{K}_\phi(A)$.

Lemma 5 ($\mathcal{K}_\phi$ with smooth ϕ)**.** *Let $A: \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be a symmetric positive semi-definite kernel and $\phi: \mathbb{R} \to \mathbb{R}$. We suppose that ϕ is an activation function following Assumptions 4 and 5; in particular, ϕ is smooth.*

Let $y \in \mathbb{R}^n$ and U be an open subset of $\mathbb{R}^n$ such that $\boldsymbol{x} \mapsto A(\boldsymbol{x}, \boldsymbol{x})$ and $\boldsymbol{x} \mapsto A(\boldsymbol{x}, \boldsymbol{y})$ are infinitely differentiable over U. Then, $\boldsymbol{x} \mapsto \mathcal{K}_\phi(A)(\boldsymbol{x}, \boldsymbol{x})$ and $\boldsymbol{x} \mapsto \mathcal{K}_\phi(A)(\boldsymbol{x}, \boldsymbol{y})$ are infinitely differentiable over U as well.

Proof. In order to prove the smoothness results over the open set U, it suffices to prove them on any open bounded subset of U. Let then $V \subseteq U$ be an open bounded set. Without loss of generality, we can assume that its closure $\operatorname{cl} V$ is also included in U.

We define B_1 and B_2 from Equation (D.46) as follows, for all $\boldsymbol{x} \in V$:

$$\begin{aligned} B_1(\boldsymbol{x}) &\triangleq \mathcal{K}_\phi(A)(\boldsymbol{x}, \boldsymbol{y}) = \mathbb{E}_{f\sim\mathcal{GP}(0,A)}\Big[\phi\big(f(\boldsymbol{x})\big)\phi\big(f(\boldsymbol{y})\big)\Big], \\ B_2(\boldsymbol{x}) &\triangleq \mathcal{K}_\phi(A)(\boldsymbol{x}, \boldsymbol{x}) = \mathbb{E}_{f\sim\mathcal{GP}(0,A)}\Big[\phi\big(f(\boldsymbol{x})\big)^2\Big]. \end{aligned} \tag{D.47}$$

In the previous expressions, Lemma 4 tells us that we can take f to be $\mathcal{C}^\infty$ over $\operatorname{cl} V$ with probability one. Hence, B_1 and B_2 are expectations of smooth functions over V. We seek to apply the dominated convergence theorem to prove that B_1 and B_2 are, in turn, smooth over V. To this end, we prove in the following the integrability of the derivatives of their integrands.

Let $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$. Using the usual notations for multi-indexed partial derivatives, via a multivariate Faà di Bruno formula (Leipnik and Pearce, 2007), we can write the derivatives $\partial^{\boldsymbol{\alpha}}(\psi \circ f)$ at $\boldsymbol{x} \in V$ for $\psi \in \{\phi, \phi^2\}$ as a weighted sum of terms of the form:

$$\psi^{(j)}\big(f(\boldsymbol{x})\big) g_1(\boldsymbol{x}) \cdots g_N(\boldsymbol{x}), \tag{D.48}$$

where the g_is are partial derivatives of f at $\boldsymbol{x}$. As A is $\mathcal{C}^\infty$ over V, each of the g_is is thus a GP with a $\mathcal{C}^\infty$ covariance function by Lemma 4. We can also write for all $\boldsymbol{x} \in V$:

$$\begin{aligned} \Big|\psi^{(j)}\big(f(\boldsymbol{x})\big) g_1(\boldsymbol{x}) \cdots g_N(\boldsymbol{x})\Big| &\leq \sup_{\boldsymbol{z}\in\operatorname{cl} V} \Big|\psi^{(j)}\big(f(\boldsymbol{z})\big) g_1(\boldsymbol{z}) \cdots g_N(\boldsymbol{z})\Big| \\ &\leq \sup_{\boldsymbol{z}_0\in\operatorname{cl} V} \Big|\psi^{(j)}\big(f(\boldsymbol{z}_0)\big)\Big| \sup_{\boldsymbol{z}_1\in\operatorname{cl} V} \big|g_1(\boldsymbol{z}_1)\big| \cdots \sup_{\boldsymbol{z}_N\in\operatorname{cl} V} \big|g_N(\boldsymbol{z}_N)\big|. \end{aligned} \tag{D.49}$$

For each i, because the covariance function of g_i is smooth over the compact set $\operatorname{cl} V$, its variance admits a maximum in $\operatorname{cl} V$ and we take σ_i^2 the double of its value. We then know from Adler (1990), that there is an M_i such that:

$$\forall m \in \mathbb{N}, \mathbb{E}_{f\sim\mathcal{GP}(0,A)}\left[\sup_{\boldsymbol{z}_i\in\operatorname{cl} V} \big|g_i(\boldsymbol{z}_i)\big|^m\right] \leq M_i^m \mathbb{E}|Y_i|^m, \tag{D.50}$$

where Y_i is a Gaussian distribution which variance is σ_i^2, the right-hand side thus being finite.

We also have, by Assumption 4 from Appendix D.1.1, that:

$$\sup_{\boldsymbol{z}\in\operatorname{cl} V} \Big|\phi^{(j)}\big(f(\boldsymbol{z})\big)\Big|^2 \leq \sup_{\boldsymbol{z}\in\operatorname{cl} V} \Big(\lambda_1^{(j)}\big|f(\boldsymbol{z})\big| + \lambda_2^{(j)}\Big)^2, \tag{D.51}$$

which is shown to be integrable over f by the same arguments as for the g_is. Moreover, the Faà di Bruno formula decomposes $\psi^{(j)}$ when $\psi = \phi^2$ as a weighted sum of terms

of the form $\phi^{(l)}\phi^{(l')}$ with $l, l' \in \mathbb{N}$. Therefore, thanks to similar arguments, for any $\psi \in \{\phi, \phi^2\}$:

$$\mathbb{E}_{f\sim\mathcal{GP}(0,A)}\left[\sup_{\boldsymbol{z}\in\operatorname{cl} V}\left|\psi^{(j)}\big(f(\boldsymbol{z})\big)\right|^2\right] < \infty. \tag{D.52}$$

Now, by using the Cauchy-Schwarz inequality, we have that:

$$\begin{aligned}&\mathbb{E}\left[\sup_{\boldsymbol{z}_0\in\operatorname{cl} V}\left|\psi^{(j)}\big(f(\boldsymbol{z}_0)\big)\right|\sup_{\boldsymbol{z}_1\in\operatorname{cl} V}\left|g_1(\boldsymbol{z}_1)\right|\cdots\sup_{\boldsymbol{z}_N\in\operatorname{cl} V}\left|g_N(\boldsymbol{z}_N)\right|\right]\\ &\quad\leq\sqrt{\mathbb{E}\left[\sup_{\boldsymbol{z}_0\in\operatorname{cl} V}\left|\psi^{(j)}\big(f(\boldsymbol{z}_0)\big)\right|^2\right]}\sqrt{\mathbb{E}\left[\sup_{\boldsymbol{z}_1\in\operatorname{cl} V}\left|g_1(\boldsymbol{z}_1)\right|^2\cdots\sup_{\boldsymbol{z}_N\in\operatorname{cl} V}\left|g_N(\boldsymbol{z}_N)\right|^2\right]}.\end{aligned} \tag{D.53}$$

By iterated applications of the Cauchy-Schwarz inequality and using the previous arguments, we can then show that:

$$\sup_{\boldsymbol{z}_0\in\operatorname{cl} V}\left|\psi^{(j)}\big(f(\boldsymbol{z}_0)\big)\right|\sup_{\boldsymbol{z}_1\in\operatorname{cl} V}\left|g_1(\boldsymbol{z}_1)\right|\cdots\sup_{\boldsymbol{z}_N\in\operatorname{cl} V}\left|g_N(\boldsymbol{z}_N)\right| \tag{D.54}$$

is integrable over f. Additionally, note that by the same arguments for the case of $\psi = \phi$, a multiplication by $\phi\big(f(\boldsymbol{y})\big)$ preserves this integrability.

We can then write for all $x \in V$, by a standard corollary of the dominated convergence theorem:

$$\begin{aligned}\partial^{\boldsymbol{\alpha}} B_1(\boldsymbol{x}) &= \mathbb{E}_{f\sim\mathcal{GP}(0,A)}\Big[\partial^{\boldsymbol{\alpha}}\left(\phi\circ f\right)\big|_{\boldsymbol{x}}\phi\big(f(\boldsymbol{y})\big)\Big],\\ \partial^{\boldsymbol{\alpha}} B_2(\boldsymbol{x}) &= \mathbb{E}_{f\sim\mathcal{GP}(0,A)}\left[\partial^{\boldsymbol{\alpha}}\left(\phi^2\circ f\right)\Big|_{\boldsymbol{x}}\right],\end{aligned} \tag{D.55}$$

which shows that B_1 and B_2 are $\mathcal{C}^\infty$ over V. This in turn means that B_1 and B_2 are $\mathcal{C}^\infty$ over U. □

Lemma 6 ($\mathcal{K}_\phi$ with piecewise smooth ϕ)**.** *Let $A\colon \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be a symmetric positive semi-definite kernel and $\phi\colon \mathbb{R} \to \mathbb{R}$. We suppose that ϕ is an activation function following Assumptions 4 and 6 (cf. Appendix D.1.1). Let us define the matrix $\Sigma_A^{\boldsymbol{x},\boldsymbol{y}}$ as:*

$$\Sigma_A^{\boldsymbol{x},\boldsymbol{y}} \triangleq \begin{pmatrix} A(\boldsymbol{x},\boldsymbol{x}) & A(\boldsymbol{x},\boldsymbol{y}) \\ A(\boldsymbol{x},\boldsymbol{y}) & A(\boldsymbol{y},\boldsymbol{y})\end{pmatrix}. \tag{D.56}$$

Let $y \in \mathbb{R}^n$ and U be an open subset of $\mathbb{R}^n$ such that $\boldsymbol{x} \mapsto A(\boldsymbol{x},\boldsymbol{x})$ and $\boldsymbol{x} \mapsto A(\boldsymbol{x},\boldsymbol{y})$ are infinitely differentiable over U. Then, $\boldsymbol{x} \mapsto \mathcal{K}_\phi(A)(\boldsymbol{x},\boldsymbol{x})$ and $\boldsymbol{x} \mapsto \mathcal{K}_\phi(A)(\boldsymbol{x},\boldsymbol{y})$ are infinitely differentiable over $U' \triangleq \big\{x \in U \;\big|\; \Sigma_A^{\boldsymbol{x},\boldsymbol{y}}$ is invertible$\big\}$.

Proof. Since $\det\Sigma_A^{\boldsymbol{x},\boldsymbol{y}}$ is smooth over U and $U' = \big\{x \in U \;\big|\; \det\Sigma_A^{\boldsymbol{x},\boldsymbol{y}} > 0\big\}$, U' is an open subset of U. Hence, similarly to the proof of Lemma 5, it suffices to prove the smoothness of B_1 and B_2 defined in Equation (D.47) on any open bounded subset of U'. Let then $V \subseteq \mathbb{R}^n$ be an open bounded set such that $\operatorname{cl} V \subseteq U'$. Note that $\det\Sigma_A^{\boldsymbol{x},\boldsymbol{y}} > 0$ implies that $A(\boldsymbol{x},\boldsymbol{x}) > 0$ and $A(\boldsymbol{y},\boldsymbol{y}) > 0$.

We will conduct in the following the proof that B_1 is smooth over V. Like in the proof of Lemma 5, the smoothness of B_2 follows the same reasoning with little adaptation; in particular, it relies on the fact that $A(\boldsymbol{x}, \boldsymbol{x}) > 0$ for all $x \in U'$, making its square root smooth over $\operatorname{cl} V$.

Since the dominated convergence theorem cannot be directly applied from Equation (D.47) because of ϕ's potential non-differentiability points D, let us decompose its expression for all $\boldsymbol{x} \in U'$:

$$B_1(\boldsymbol{x}) = \mathbb{E}_{f \sim \mathcal{GP}(0,A)}\Big[\phi\big(f(\boldsymbol{x})\big)\phi\big(f(\boldsymbol{y})\big)\Big] = \mathbb{E}_{(z,z') \sim \mathcal{N}\left((0,0), \Sigma_A^{\boldsymbol{x},\boldsymbol{y}}\right)}\Big[\phi(z)\phi\big(z'\big)\Big] \tag{D.57}$$

$$= \mathbb{E}_{z' \sim \mathcal{N}\left(0, A(\boldsymbol{y},\boldsymbol{y})\right)}\left[\phi\big(z'\big)\mathbb{E}_{z \sim \mathcal{N}\left(\frac{A(\boldsymbol{x},\boldsymbol{y})}{A(\boldsymbol{y},\boldsymbol{y})}z', A(\boldsymbol{x},\boldsymbol{x}) - \frac{A(\boldsymbol{x},\boldsymbol{y})^2}{A(\boldsymbol{y},\boldsymbol{y})}\right)}\big[\phi(z)\big]\right] \tag{D.58}$$

$$= \mathbb{E}_{z' \sim \mathcal{N}\left(0, A(\boldsymbol{y},\boldsymbol{y})\right)}\Big[\phi\big(z'\big)h\big(z', \boldsymbol{x}\big)\Big], \tag{D.59}$$

where h is defined as:

$$h\big(z', \boldsymbol{x}\big) \triangleq \int_{-\infty}^{+\infty} \phi(z) \cdot \frac{1}{\sigma(\boldsymbol{x})\sqrt{2\pi}} \mathrm{e}^{-\frac{1}{2}\left(\frac{z-\mu(\boldsymbol{x})z'}{\sigma(\boldsymbol{x})}\right)^2} \mathrm{d}z, \tag{D.60}$$

and:

$$\mu(\boldsymbol{x}) = \frac{A(\boldsymbol{x},\boldsymbol{y})}{A(\boldsymbol{y},\boldsymbol{y})}, \qquad \sigma(\boldsymbol{x}) = \sqrt{\frac{\det \Sigma_A^{\boldsymbol{x},\boldsymbol{y}}}{A(\boldsymbol{y},\boldsymbol{y})}}. \tag{D.61}$$

Now, if $D = \{c_1, \ldots, c_L\}$ with $L \in \mathbb{N}$ and $c_1 < \cdots < c_L$, the c_ls constitute the non-differentiability points of ϕ; we can then decompose the integration of ϕ in Equation (D.60) as a sum of $L+1$ integrals with differentiable integrands, using $c_0 = -\infty$ and $c_{L+1} = +\infty$:

$$h(\varepsilon, \boldsymbol{x}) = \frac{1}{\sqrt{2\pi}} \sum_{l=0}^{L} \int_{c_l}^{c_{l+1}} \frac{\phi(z)}{\sigma(\boldsymbol{x})} \mathrm{e}^{-\frac{1}{2}\left(\frac{z-\mu(\boldsymbol{x})z'}{\sigma(\boldsymbol{x})}\right)^2} \mathrm{d}z. \tag{D.62}$$

Therefore, it remains to show the smoothness of all applications $B_{1,l}$ for $l \in [\![0, L]\!]$ defined as:

$$B_{1,j}(\boldsymbol{x}) = \mathbb{E}_{z' \sim \mathcal{N}\left(0, A(\boldsymbol{y},\boldsymbol{y})\right)}\left[\int_{c_l}^{c_{l+1}} \frac{\phi\big(z'\big)\phi(z)}{\sigma(\boldsymbol{x})\sqrt{2\pi}} \mathrm{e}^{-\frac{1}{2}\left(\frac{z-\mu(\boldsymbol{x})z'}{\sigma(\boldsymbol{x})}\right)^2} \mathrm{d}z\right]. \tag{D.63}$$

The rest of this proof unfolds similarly to the one of Lemma 5. Indeed, the integrand of Equation (D.63) is smooth over $\operatorname{cl} V$. There remains to show that all derivatives of this integrand are dominated by an integrable function of z and z'. Consider the following integrand:

$$\iota\big(z, z', \boldsymbol{x}\big) = \frac{\phi\big(z'\big)\phi(z)}{\sigma(\boldsymbol{x})\sqrt{2\pi}} \mathrm{e}^{-\frac{1}{2}\left(\frac{z-\mu(\boldsymbol{x})z'}{\sigma(\boldsymbol{x})}\right)^2}. \tag{D.64}$$

By applying the multivariate Faà di Bruno formula and noticing that σ and μ are smooth over the closed set $\operatorname{cl} V$, we know that the derivatives of $\iota\big(z, z', \boldsymbol{x}\big)$ with respect to $\boldsymbol{x}$ for any derivation order are weighted sums of terms of the form:

$$z^k z'^{k'} \kappa(\boldsymbol{x}) \frac{\phi\big(z'\big)\phi(z)}{\sigma(\boldsymbol{x})\sqrt{2\pi}} \mathrm{e}^{-\frac{1}{2}\left(\frac{z-\mu(\boldsymbol{x})z'}{\sigma(\boldsymbol{x})}\right)^2}, \tag{D.65}$$

where κ is a smooth function over $\operatorname{cl} V$ and $k, k' \in \mathbb{N}$. Moreover, because σ, μ and κ are smooth over the closed set $\operatorname{cl} V$ with positive values for σ, there are constants a_1, a_2 and a_3 such that:

$$\left| z^k z'^{k'} \kappa(\boldsymbol{x}) \frac{\phi\big(z'\big)\phi(z)}{\sigma(\boldsymbol{x})\sqrt{2\pi}} \mathrm{e}^{-\frac{1}{2}\left(\frac{z-\mu(\boldsymbol{x})z'}{\sigma(\boldsymbol{x})}\right)^2} \right| \leq \left| z^k z'^{k'} \phi\big(z'\big)\phi(z) \right| a_3 \mathrm{e}^{-\frac{1}{2}\left(\frac{z-a_1 z'}{a_2}\right)^2}, \tag{D.66}$$

which is integrable over z via Assumption 4 and Equation (D.1). Finally, let us notice that for some constants b_1, b_2 and b_3:

$$\int_{c_l}^{c_{l+1}} \left| z^k z'^{k'} \phi\big(z'\big)\phi(z) \right| a_3 \mathrm{e}^{-\frac{1}{2}\left(\frac{z-a_1 z'}{a_2}\right)^2} \leq b_1 \mathbb{E}_{z\sim\mathcal{N}(b_2 z', b_3)} \left| z^k z'^{k'} \phi\big(z'\big)\phi(z) \right|, \tag{D.67}$$

which is also integrable with respect to $\mathbb{E}_{z'\sim\mathcal{N}\left(0, A(\boldsymbol{y},\boldsymbol{y})\right)}$ by similar arguments (see also the integrability of Equation (D.50) in Lemma 5). This concludes the proof of integrability required to apply the dominated convergence theorem, allowing us to conclude about the smoothness of all $B_{1,j}$ and, in turn, of B_1 over U'. □

Remark 5 (Relaxed condition for smoothness)**.** The invertibility condition of Lemma 6 is actually stronger than needed: it suffices to assume that the rank of $\Sigma_A^{\boldsymbol{x},\boldsymbol{y}}$ remains constant in a neighborhood of $\boldsymbol{x}$.

D.1.3.2. Differentiability of Conjugate Kernels, NTKs and Discriminators

From the previous lemmas, we can then prove the results of Section 6.4.3. We start by demonstrating the smoothness of the conjugate kernel for dense networks, and conclude in consequence about the smoothness of the NTK and trained network.

Lemma 7 (Differentiability of the conjugate kernel)**.** *Let k_{c} be the conjugate kernel (J. Lee, Bahri, et al., 2018) of an infinite-width dense non-residual architecture such as in Assumption 4. For any $\boldsymbol{y} \in \mathbb{R}^n$, the following holds for $A \in \big\{k_{\mathrm{c}}, \mathcal{K}_{\phi'}(k_{\mathrm{c}})\big\}$:*

- *if Assumption 5 holds, then $\boldsymbol{x} \mapsto A(\boldsymbol{x}, \boldsymbol{x})$ and $\boldsymbol{x} \mapsto A(\boldsymbol{x}, \boldsymbol{y})$ are smooth everywhere over $\mathbb{R}^n$;*
- *if Assumption 6 holds, then $\boldsymbol{x} \mapsto A(\boldsymbol{x}, \boldsymbol{x})$ and $\boldsymbol{x} \mapsto A(\boldsymbol{x}, \boldsymbol{y})$ are smooth over an open set whose complement has null Lebesgue measure.*

Proof. We define the following kernel:

$$C_L^\phi(\boldsymbol{x},\boldsymbol{y}) = \mathbb{E}_{f\sim\mathcal{GP}\left(0,C_{L-1}^\phi\right)}\Big[\phi\big(f(\boldsymbol{x})\big)\phi\big(f(\boldsymbol{y})\big)\Big] + \beta^2 = \mathcal{K}_\phi\Big(C_{L-1}^\phi\Big) + \beta^2, \tag{D.68}$$

with:

$$C_0^\phi(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{n}\boldsymbol{x}^\top\boldsymbol{y} + \beta^2. \tag{D.69}$$

We have that $k_\mathrm{c} = C_L^\phi$, with L being the number of hidden layers in the network. Therefore, Lemma 5 ensures the smoothness result under Assumption 5.

Let us now consider Assumption 6 (cf. the detailed assumption in Appendix D.1.1); in particular, $\beta > 0$. We prove by induction over L in the following that:

- $B_1 \colon \boldsymbol{x} \mapsto C_L^\phi(\boldsymbol{x},\boldsymbol{y})$ is smooth over $U = \big\{x \in \mathbb{R}^n \;\big|\; \|\boldsymbol{x}\| \neq \|\boldsymbol{y}\|\big\}$;
- $B_2 \colon \boldsymbol{x} \mapsto C_L^\phi(\boldsymbol{x},\boldsymbol{x})$ is smooth;
- for all $\boldsymbol{x},\boldsymbol{x}' \in \mathbb{R}^n$ with $\|\boldsymbol{x}\| \neq \big\|\boldsymbol{x}'\big\|$, $B_2(\boldsymbol{x}) \neq B_2\big(\boldsymbol{x}'\big)$.

The result is immediate for $L = 0$. We now suppose that it holds for some $L \in \mathbb{N}$ and prove that it also holds for $L+1$ hidden layers. Let us express B_2:

$$B_2(\boldsymbol{x}) = \mathbb{E}_{\varepsilon\sim\mathcal{N}(0,1)}\left[\phi\left(\varepsilon\sqrt{C_L^\phi(\boldsymbol{x},\boldsymbol{x}) + \beta^2}\right)^2\right]. \tag{D.70}$$

Using Lemma 6 and Remark 5, the fact that $\beta > 0$ and the induction hypothesis ensures that B_2 is smooth. Moreover, Assumption 6, in particular Equation (D.2), allows us to assert that $\|\boldsymbol{x}\| \neq \big\|\boldsymbol{x}'\big\|$ implies $B_2(\boldsymbol{x}) \neq B_2\big(\boldsymbol{x}'\big)$.

Finally, in order to apply Lemma 6 to prove the smoothness of B_1 over U, there remains to show that the following matrix is invertible:

$$\Sigma_\beta^{\boldsymbol{x},\boldsymbol{y}} \triangleq \begin{pmatrix} C_L^\phi(\boldsymbol{x},\boldsymbol{x}) + \beta^2 & C_L^\phi(\boldsymbol{x},\boldsymbol{y}) + \beta^2 \\ C_L^\phi(\boldsymbol{x},\boldsymbol{y}) + \beta^2 & C_L^\phi(\boldsymbol{y},\boldsymbol{y}) + \beta^2 \end{pmatrix}. \tag{D.71}$$

Let us compute its determinant:

$$\begin{aligned} \det\Sigma_\beta^{\boldsymbol{x},\boldsymbol{y}} &= \Big(C_L^\phi(\boldsymbol{x},\boldsymbol{x}) + \beta^2\Big)\Big(C_L^\phi(\boldsymbol{y},\boldsymbol{y}) + \beta^2\Big) - \Big(C_L^\phi(\boldsymbol{x},\boldsymbol{y}) + \beta^2\Big)^2 \\ &= \det\Sigma_0^{\boldsymbol{x},\boldsymbol{y}} + \beta^2\Big(C_L^\phi(\boldsymbol{x},\boldsymbol{x}) + C_L^\phi(\boldsymbol{y},\boldsymbol{y}) - 2C_L^\phi(\boldsymbol{x},\boldsymbol{y})\Big). \end{aligned} \tag{D.72}$$

C_L^ϕ is a symmetric positive semi-definite kernel, thus:

$$\det\Sigma_\beta^{\boldsymbol{x},\boldsymbol{y}} - \det\Sigma_0^{\boldsymbol{x},\boldsymbol{y}} = \beta^2 \cdot \begin{pmatrix} 1 & -1 \end{pmatrix} \Sigma_0^{\boldsymbol{x},\boldsymbol{y}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} \geq 0. \tag{D.73}$$

Hence, if $\det\Sigma_0^{\boldsymbol{x},\boldsymbol{y}} > 0$, then $\det\Sigma_\beta^{\boldsymbol{x},\boldsymbol{y}} > 0$. Besides, if $\det\Sigma_0^{\boldsymbol{x},\boldsymbol{y}} = 0$, then:

$$\det\Sigma_\beta^{\boldsymbol{x},\boldsymbol{y}} = \beta^2\Big(\sqrt{B_2(\boldsymbol{x})} - \sqrt{B_2(\boldsymbol{y})}\Big)^2 > 0, \tag{D.74}$$

for all $x \in U$. This proves that B_1 is indeed smooth over U, and concludes the induction.

Note that U is indeed an open set whose complement in $\mathbb{R}^n$ has null Lebesgue measure. Overall, the result is thus proved for $A = k_c$; a similar reasoning using the previous induction result also transfers the result to $A = \mathcal{K}_{\phi'}(k_c)$. □

Proposition 2 (Differentiability of k)**.** Let k be the NTK of an infinite-width architecture following Assumption 4. For any $\boldsymbol{y} \in \mathbb{R}^n$:

- if Assumption 5 holds, then $k(\cdot, \boldsymbol{y})$ is smooth everywhere over $\mathbb{R}^n$;
- if Assumption 6 holds, then $k(\cdot, \boldsymbol{y})$ is smooth almost everywhere over $\mathbb{R}^n$, in particular over an open set whose complement has null Lebesgue measure.

Proof. According to the definitions of Jacot, Gabriel, and Hongler (2018), Arora, Du, W. Hu, et al. (2019), and K. Huang et al. (2020), the smoothness of the kernel is guaranteed whenever the conjugate kernel k_c and its transform $\mathcal{K}_{\phi'}(k_c)$ are smooth; the result of Lemma 7 then applies. In the case of residual networks, there is a slight adaptation of the formula which does not change its regularity. Regarding convolutional networks, their conjugate kernels and NTKs involve finite combinations of such dense conjugate kernels and NTKs, thereby preserving their smoothness almost everywhere. □

Theorem 2 (Differentiability of f_t)**.** Let f_t be a solution to Equation (6.9) under Assumptions 1 and 3 by Theorem 1, with k the NTK of an infinite-width neural network and f_0 an initialization of the latter.

Then, under Assumptions 4 and 5, f_t is smooth everywhere. Under Assumptions 4 and 6, f_t is smooth almost everywhere, in particular over an open set whose complement has null Lebesgue measure.

Proof. From Theorem 1, we have:

$$f_t - f_0 = \mathcal{T}_{k,\hat{\gamma}}\left(\int_0^t \nabla^{\hat{\gamma}} \mathcal{L}_{\hat{\alpha}}(f_s)\,\mathrm{d}s\right). \tag{D.75}$$

We observe that $\mathcal{T}_{k,\hat{\gamma}}(h)$ has, for any $h \in L^2(\hat{\gamma})$, a regularity which only depends on the regularity of $k(\cdot, \boldsymbol{y})$ for $\boldsymbol{y} \in \operatorname{supp} \hat{\gamma}$. Indeed, if $k(\cdot, \boldsymbol{y})$ is smooth in a certain neighborhood V for every such $\boldsymbol{y}$, we can bound $\partial^{\boldsymbol{\alpha}} k(\cdot, \boldsymbol{y})$ over V for every $\boldsymbol{y}$ and any multi-index $\boldsymbol{\alpha}$ and then use dominated convergence to prove that $\mathcal{T}_{k,\hat{\gamma}}(h)(\cdot)$ is smooth over V. Therefore, the regularity of $k(\cdot, \boldsymbol{y})$ transfers to $f_t - f_0$. Given Proposition 2, there remains to prove the same result for f_0.

The theorem then follows from the fact that f_0 has the same regularity as its conjugate kernel k_c thanks to Lemma 4 because f_0 is a sample from the GP with kernel k_c. Lemma 7 shows the smoothness almost everywhere over an open set of applications $\boldsymbol{x} \mapsto k_c(\boldsymbol{x}, \boldsymbol{y})$; to apply Lemma 4 and concludes this proof, this result must be generalized to prove the smoothness of k_c with respect to both its inputs. This can be done by generalizing the proofs of Lemmas 5 and 6 to show the smoothness of kernels with respect to both $\boldsymbol{x}$ and $\boldsymbol{y}$, with the same arguments than for $\boldsymbol{x}$ alone. □

Remark 6. In the previous theorem, f_0 is considered to be the initialization of the network. However, we highlight that, without loss of generality, this theorem encompasses the change of training distribution $\hat{\gamma}$ during GAN training. Indeed, as explained in Section 6.4.1, f_0 after j steps of generator training can actually be decomposed as, for some $h_k \in L^2(\hat{\gamma}_k)$, $k \in [\![1, j]\!]$:

$$f_0 = f^0 + \sum_{k=1}^{j} \mathcal{T}_{k,\hat{\gamma}_k}(h_k), \tag{D.76}$$

by taking into account the updates of the discriminators over the whole GAN optimization process. The proof of Theorem 2 can then be applied similarly in this case by showing the differentiability of $f_0 - f^0$ on the one hand and of f^0, being the initialization of the discriminator at the very beginning of GAN training, on the other hand.

D.1.4. Dynamics of the Generated Distribution

We derive in this proposition the differential equation governing the dynamics of the generated distribution.

Proposition 3 (Dynamics of α_ℓ)**.** Under Assumptions 4 and 5, Equation (6.3) is well-posed. Let us consider its continuous-time version with discriminators trained on discrete distributions as described above:

$$\partial_\ell \theta_\ell = -\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}\left[\nabla_\theta g_\ell(\boldsymbol{z})^\top \nabla_{\boldsymbol{x}}\, c_{f^\star_{\hat{\alpha}_{g_\ell}}}(\boldsymbol{x})\Big|_{\boldsymbol{x}=g_\ell(\boldsymbol{z})}\right]. \tag{D.77}$$

This yields, with k_{g_ℓ} the NTK of the generator g_ℓ:

$$\partial_\ell g_\ell = -\mathcal{T}_{k_{g_\ell},p_{\boldsymbol{z}}}\left(\boldsymbol{z} \mapsto \nabla_{\boldsymbol{x}}\, c_{f^\star_{\hat{\alpha}_{g_\ell}}}(\boldsymbol{x})\Big|_{\boldsymbol{x}=g_\ell(\boldsymbol{z})}\right). \tag{D.78}$$

Equivalently, the following continuity equation holds for the joint distribution $\alpha_\ell^{\boldsymbol{z}} \triangleq (\mathrm{id}, g_\ell)_\sharp p_{\boldsymbol{z}}$:

$$\partial_\ell \alpha_\ell^{\boldsymbol{z}} = -\nabla_{\boldsymbol{x}} \cdot \left(\alpha_\ell^{\boldsymbol{z}} \mathcal{T}_{k_{g_\ell},p_{\boldsymbol{z}}}\left(\boldsymbol{z} \mapsto \nabla_{\boldsymbol{x}}\, c_{f^\star_{\hat{\alpha}_{g_\ell}}}(\boldsymbol{x})\Big|_{\boldsymbol{x}=g_\ell(\boldsymbol{z})}\right)\right). \tag{D.79}$$

Proof. Assumptions 4 and 5 ensure, via Proposition 2 and Theorem 2 that the trained discriminator is differentiable everywhere at all times, whatever the state of the generator. Therefore, Equation (6.3) is well-posed.

By following Mroueh, Sercu, and Raj (2019, Equation (5))'s reasoning on a similar equation, Equation (D.77) yields the following generator dynamics for all inputs $\boldsymbol{z} \in \mathbb{R}^d$:

$$\partial_\ell g_\ell(\boldsymbol{z}) = -\mathbb{E}_{\boldsymbol{z}'\sim p_{\boldsymbol{z}}}\left[\nabla_{\theta_\ell} g_\ell(\boldsymbol{z})^\top \nabla_{\theta_\ell} g_\ell(\boldsymbol{z}') \nabla_{\boldsymbol{x}}\, c_{f^\star_{\hat{\alpha}_{g_\ell}}}(\boldsymbol{x})\Big|_{\boldsymbol{x}=g_\ell(\boldsymbol{z}')}\right]. \tag{D.80}$$

We recognize the NTK k_{g_ℓ} of the generator as:

$$k_{g_\ell}(\boldsymbol{z}, \boldsymbol{z}') \triangleq \nabla_{\theta_\ell} g_\ell(\boldsymbol{z})^\top \nabla_{\theta_\ell} g_\ell(\boldsymbol{z}'). \tag{D.81}$$

From this, we obtain the dynamics of the generator:

$$\partial_\ell g_\ell = -\mathcal{T}_{k_{g_\ell}, p_{\boldsymbol{z}}} \left(\boldsymbol{z} \mapsto \nabla_{\boldsymbol{x}} c_{f^\star_{\hat{\alpha}} g_\ell}(\boldsymbol{x}) \Big|_{\boldsymbol{x}=g_\ell(\boldsymbol{z})} \right). \tag{D.82}$$

In other words, the transported particles $\big(\boldsymbol{z}, g_\ell(\boldsymbol{z})\big)$ have trajectories X_ℓ which are solutions of the ODE:

$$\frac{\mathrm{d}X_\ell}{\mathrm{d}\ell} = \big(\mathbf{0}, v_\ell(X_\ell)\big), \tag{D.83}$$

where:

$$v_\ell = -\mathcal{T}_{k_{g_\ell}, p_{\boldsymbol{z}}} \left(\boldsymbol{z} \mapsto \nabla_{\boldsymbol{x}} c_{f^\star_{\hat{\alpha}} g_\ell}(\boldsymbol{x}) \Big|_{\boldsymbol{x}=g_\ell(\boldsymbol{z})} \right). \tag{D.84}$$

Then, because $\alpha^{\boldsymbol{z}}_\ell \triangleq (\mathrm{id}, g_\ell)_\sharp p_{\boldsymbol{z}}$ is the induced transported density, following Ambrosio and Crippa (2014), whenever the ODE above is well-defined and has unique solutions (which is necessarily the case for any trained g), $\alpha^{\boldsymbol{z}}_\ell$ verifies the continuity equation with the velocity field v_ℓ:

$$\begin{aligned} \partial_\ell \alpha^{\boldsymbol{z}}_\ell &= -\nabla_{\boldsymbol{z},\boldsymbol{x}} \cdot \left(\alpha^{\boldsymbol{z}}_\ell \left(\mathbf{0}, \mathcal{T}_{k_{g_\ell}, p_{\boldsymbol{z}}} \left(\boldsymbol{z} \mapsto \nabla_{\boldsymbol{x}} c_{f^\star_{\hat{\alpha}} g_\ell}(\boldsymbol{x}) \Big|_{\boldsymbol{x}=g_\ell(\boldsymbol{z})} \right) \right) \right) \\ &= -\nabla_{\boldsymbol{x}} \cdot \left(\alpha^{\boldsymbol{z}}_\ell \mathcal{T}_{k_{g_\ell}, p_{\boldsymbol{z}}} \left(\boldsymbol{z} \mapsto \nabla_{\boldsymbol{x}} c_{f^\star_{\hat{\alpha}} g_\ell}(\boldsymbol{x}) \Big|_{\boldsymbol{x}=g_\ell(\boldsymbol{z})} \right) \right). \end{aligned} \tag{D.85}$$

This yields the desired result. □

D.1.5. Optimality in Concave Setting

We derive an optimality result for concave bounded loss functions of the discriminator and positive definite kernels.

D.1.5.1. Assumptions

We first assume that the NTK is positive definite over the training dataset.

Assumption 7 (Positive definite kernel)**.** *k is positive definite over $\hat{\gamma}$.*

This positive definiteness property equates for finite datasets to the invertibility of the mapping

$$\begin{aligned} \mathcal{T}_{k,\hat{\gamma}}\big|_{\operatorname{supp}\hat{\gamma}} \colon L^2(\hat{\gamma}) &\to L^2(\hat{\gamma}) \\ h &\mapsto \mathcal{T}_{k,\hat{\gamma}}(h)\big|_{\operatorname{supp}\hat{\gamma}} \end{aligned}, \tag{D.86}$$

that can be seen as a multiplication by the invertible Gram matrix of k over $\hat{\gamma}$. We further discuss this hypothesis in Appendix D.2.5.

We also assume the following properties on the discriminator loss function.

Assumption 8 (Concave loss)**.** *$\mathcal{L}_{\hat{\alpha}}$ is concave and bounded from above, and its supremum is reached on a unique point $y^\star$ in $L^2(\hat{\gamma})$.*

Moreover, we need for the sake of the proof a uniform continuity assumption on the solution to Equation (6.9).

Assumption 9 (Solution continuity)**.** *$t \mapsto f_t|_{\operatorname{supp}\hat{\gamma}}$ is uniformly continuous over $\mathbb{R}_+$.*

Note that these assumptions are verified in the case of LSGAN, which is the typical application of the optimality results that we prove in the following.

D.1.5.2. Optimality Result

Proposition 7 (Asymptotic optimality)**.** *Under Assumptions 1 to 3 and 7 to 9, f_t converges pointwise when $t \to \infty$, and:*

$$\mathcal{L}_{\hat{\alpha}}(f_t) \xrightarrow[t\to\infty]{} \mathcal{L}_{\hat{\alpha}}(y^\star), \qquad f_\infty = f_0 + \mathcal{T}_{k,\hat{\gamma}}\left(\mathcal{T}_{k,\hat{\gamma}}|^{-1}_{\operatorname{supp}\hat{\gamma}}\left(y^\star - f_0|_{\operatorname{supp}\hat{\gamma}}\right)\right), \tag{D.87}$$

with:

$$f_\infty|_{\operatorname{supp}\hat{\gamma}} = y^\star = \operatorname*{arg\,max}_{y\in L^2(\hat{\gamma})} \mathcal{L}_{\hat{\alpha}}(y). \tag{D.88}$$

This result ensures that, for concave losses such as LSGAN, the optimum for $\mathcal{L}_{\hat{\alpha}}$ in $L^2(\Omega)$ is reached for infinite training times by neural network training in the infinite-width regime when the NTK of the discriminator is positive definite. However, this also provides the expression of the optimal network outside $\operatorname{supp}\hat{\gamma}$ thanks to the smoothing of $\hat{\gamma}$.

In order to prove this proposition, we need the following intermediate results: the first one about the functional gradient of $\mathcal{L}_{\hat{\alpha}}$ on the solution f_t; the second one about a direct application of positive definite kernels showing that one can retrieve $f \in \mathcal{H}_k^{\hat{\gamma}}$ over all Ω from its restriction to $\operatorname{supp}\hat{\gamma}$.

Lemma 8. *Under Assumptions 1 to 3 and 7 to 9, $\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f_t) \to 0$ when $t \to \infty$. Since $\operatorname{supp}\hat{\gamma}$ is finite, this limit can be interpreted pointwise.*

Proof. Assumptions 1 to 3 ensure the existence and uniqueness of f_t, by Theorem 1.
$t \mapsto \hat{f}_t \triangleq f_t|_{\operatorname{supp}\hat{\gamma}}$ and $\mathcal{L}_{\hat{\alpha}}$ being differentiable, $t \mapsto \mathcal{L}_{\hat{\alpha}}(f_t)$ is differentiable, and:

$$\partial_t\mathcal{L}_{\hat{\alpha}}(f_t) = \left\langle \nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f_t), \partial_t\hat{f}_t\right\rangle_{L^2(\hat{\gamma})} = \left\langle \nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f_t), \mathcal{T}_{k,\hat{\gamma}}\left(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f_t)\right)\right\rangle_{L^2(\hat{\gamma})}, \tag{D.89}$$

using Equation (6.9). This equates to:

$$\partial_t\mathcal{L}_{\hat{\alpha}}(f_t) = \left\|\mathcal{T}_{k,\hat{\gamma}}\left(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f_t)\right)\right\|^2_{\mathcal{H}_k^{\hat{\gamma}}} \geq 0, \tag{D.90}$$

where $\|\cdot\|_{\mathcal{H}_k^{\hat{\gamma}}}$ is the semi-norm associated to the RKHS $\mathcal{H}_k^{\hat{\gamma}}$. Note that this semi-norm is dependent on the restriction of its input to $\operatorname{supp}\hat{\gamma}$ only. Therefore, $t \mapsto \mathcal{L}_{\hat{\alpha}}(f_t)$ is increasing. Since $\mathcal{L}_{\hat{\alpha}}$ is bounded from above, $t \mapsto \mathcal{L}_{\hat{\alpha}}(f_t)$ admits a limit when $t \to \infty$.

We now aim at proving from the latter fact that $\partial_t \mathcal{L}_{\hat{\alpha}}(f_t) \to 0$ when $t \to \infty$. We notice that $\|\cdot\|^2_{\mathcal{H}_k^{\hat{\gamma}}}$ is uniformly continuous over $L^2(\hat{\gamma})$ since $\operatorname{supp}\hat{\gamma}$ is finite, $\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}$ is uniformly continuous over $L^2(\hat{\gamma})$ since a' and b' are Lipschitz-continuous, $\mathcal{T}_{k,\hat{\gamma}}\big|_{\operatorname{supp}\hat{\gamma}}$ is uniformly continuous as it amounts to a finite matrix multiplication, and Assumption 9 gives that $t \mapsto f_t|_{\operatorname{supp}\hat{\gamma}}$ is uniformly continuous over $\mathbb{R}_+$. Therefore, their composition $t \mapsto \partial_t \mathcal{L}_{\hat{\alpha}}(f_t)$ (from Equation (D.90)) is uniformly continuous over $\mathbb{R}_+$. Using Barbălat's Lemma (Farkas and Wegner, 2016), we conclude that $\partial_t \mathcal{L}_{\hat{\alpha}}(f_t) \to 0$ when $t \to \infty$.

Furthermore, k is positive definite over $\hat{\gamma}$ by Assumption 7, so $\|\cdot\|_{\mathcal{H}_k^{\hat{\gamma}}}$ is actually a norm. Therefore, since $\operatorname{supp}\hat{\gamma}$ is finite, the following pointwise convergence holds:

$$\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f_t) \xrightarrow[t\to\infty]{} 0. \tag{D.91}$$

□

Lemma 9 ($\mathcal{H}_k^{\hat{\gamma}}$ determined by $\operatorname{supp}\hat{\gamma}$)**.** *Under Assumptions 1, 2 and 7, for all $f \in \mathcal{H}_k^{\hat{\gamma}}$, the following holds:*

$$f = \mathcal{T}_{k,\hat{\gamma}}\left(\mathcal{T}_{k,\hat{\gamma}}\big|^{-1}_{\operatorname{supp}\hat{\gamma}}\left(f|_{\operatorname{supp}\hat{\gamma}}\right)\right). \tag{D.92}$$

Proof. Since k is positive definite by Assumption 7, then $\mathcal{T}_{k,\hat{\gamma}}\big|_{\operatorname{supp}\hat{\gamma}}$ from Equation (D.86) is invertible. Let $f \in \mathcal{H}_k^{\hat{\gamma}}$. Then, by definition of the RKHS in Definition 2, there exists $h \in L^2(\hat{\gamma})$ such that $f = \mathcal{T}_{k,\hat{\gamma}}(h)$. In particular, $f|_{\operatorname{supp}\hat{\gamma}} = \mathcal{T}_{k,\hat{\gamma}}\big|_{\operatorname{supp}\hat{\gamma}}(h)$, hence $h = \mathcal{T}_{k,\hat{\gamma}}\big|^{-1}_{\operatorname{supp}\hat{\gamma}}\left(f|_{\operatorname{supp}\hat{\gamma}}\right)$. □

We can now prove the desired proposition.

Proof of Proposition 7. Let us first show that f_t converges to the optimum $y^\star$ in $L^2(\hat{\gamma})$. By applying Lemma 8, we know that $\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f_t) \to 0$ when $t \to \infty$. Given that the supremum of the differentiable concave function $\mathcal{L}_{\hat{\alpha}} \colon L^2(\hat{\gamma}) \to \mathbb{R}$ is achieved at a unique point $y^\star \in L^2(\hat{\gamma})$ with finite $\operatorname{supp}\hat{\gamma}$, then the latter convergence result implies that $\hat{f}_t \triangleq f_t|_{\operatorname{supp}\hat{\gamma}}$ converges pointwise to $y^\star$ when $t \to \infty$.

Given this convergence in $L^2(\hat{\gamma})$, we can deduce convergence on the whole domain Ω by noticing that $f_t - f_0 \in \mathcal{H}_k^{\hat{\gamma}}$, from Corollary 1. Thus, using Lemma 9:

$$f_t - f_0 = \mathcal{T}_{k,\hat{\gamma}}\left(\mathcal{T}_{k,\hat{\gamma}}\big|^{-1}_{\operatorname{supp}\hat{\gamma}}\left((f_t - f_0)\big|_{\operatorname{supp}\hat{\gamma}}\right)\right). \tag{D.93}$$

Again, since $\operatorname{supp}\hat{\gamma}$ is finite, and $\mathcal{T}_{k,\hat{\gamma}}\big|^{-1}_{\operatorname{supp}\hat{\gamma}}$ can be expressed as a matrix multiplication, the fact that f_t converges to $y^\star$ over $\operatorname{supp}\hat{\gamma}$ implies that:

$$\mathcal{T}_{k,\hat{\gamma}}\big|^{-1}_{\operatorname{supp}\hat{\gamma}}\left((f_t - f_0)\big|_{\operatorname{supp}\hat{\gamma}}\right) \xrightarrow[t\to\infty]{} \mathcal{T}_{k,\hat{\gamma}}\big|^{-1}_{\operatorname{supp}\hat{\gamma}}\left(y^\star - f_0|_{\operatorname{supp}\hat{\gamma}}\right). \tag{D.94}$$

Finally, using the definition of the integral operator in Definition 2, the latter convergence implies the following desired pointwise convergence:

$$f_t \xrightarrow[t\to\infty]{} f_0 + \mathcal{T}_{k,\hat{\gamma}}\left(\mathcal{T}_{k,\hat{\gamma}}|^{-1}_{\operatorname{supp}\hat{\gamma}}\left(y^\star - f_0|_{\operatorname{supp}\hat{\gamma}}\right)\right). \tag{D.95}$$

We showed at the beginning of this proof that f_t converges to the optimum $y^\star$ in $L^2(\hat{\gamma})$, so $\mathcal{L}_{\hat{\alpha}}(f_t) \to \mathcal{L}_{\hat{\alpha}}(y^\star)$ by continuity of $\mathcal{L}_{\hat{\alpha}}$ as claimed in the proposition. □

D.1.6. Case Studies of Discriminator Dynamics

We study in the rest of this section the expression of the discriminators in the case of the IPM loss and LSGAN, as described in Section 6.5, and of the original GAN formulation.

D.1.6.1. Preliminaries

We first need to introduce some definitions.

The presented solutions to Equation (6.9) leverage a notion of functions of linear operators, similarly to functions of matrices (Higham, 2008). We define such functions in the simplified case of non-negative symmetric compact operators with a finite number of eigenvalues, such as $\mathcal{T}_{k,\hat{\gamma}}$.

Definition 3 (Linear operator)**.** Let $\mathcal{A}\colon L^2(\hat{\gamma}) \to L^2(\Omega)$ be a non-negative symmetric compact linear operator with a finite number of eigenvalues, for which the spectral theorem guarantees the existence of a countable orthonormal basis of eigenfunctions with non-negative eigenvalues. If $\varphi\colon \mathbb{R}_+ \to \mathbb{R}$, we define $\varphi(\mathcal{A})$ as the linear operator with the same eigenspaces as $\mathcal{A}$, with their respective eigenvalues mapped by φ; in other words, if λ is an eigenvalue of $\mathcal{A}$, then $\varphi(\mathcal{A})$ admits the eigenvalue $\varphi(\lambda)$ with the same eigenspace.

In the case where $\mathcal{A}$ is a matrix, this amounts to diagonalizing $\mathcal{A}$ and transforming its diagonalization elementwise using φ. Note that $\mathcal{T}_{k,\hat{\gamma}}$ has a finite number of eigenvalues since it is generated by a finite linear combination of linear operators (see Definition 2).

We also need to define the following Radon–Nikodym derivatives with inputs in $\operatorname{supp}\hat{\gamma}$:

$$\rho = \frac{\mathrm{d}\left(\hat{\beta} - \hat{\alpha}\right)}{\mathrm{d}\left(\hat{\beta} + \hat{\alpha}\right)}, \qquad \rho_1 = \frac{\mathrm{d}\hat{\alpha}}{\mathrm{d}\hat{\gamma}}, \qquad \rho_2 = \frac{\mathrm{d}\hat{\beta}}{\mathrm{d}\hat{\gamma}}, \tag{D.96}$$

knowing that

$$\rho = \frac{1}{2}(\rho_2 - \rho_1), \qquad \rho_1 + \rho_2 = 2. \tag{D.97}$$

These functions help us to compute the functional gradient of $\mathcal{L}_{\hat{\alpha}}$, as follows.

Lemma 10 (Loss derivative)**.** *Under Assumption 3:*

$$\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f) = \rho_1 a'_f - \rho_2 b'_f = \rho_1 \cdot \big(a' \circ f\big) - \rho_2 \cdot \big(b' \circ f\big). \tag{D.98}$$

Proof. We have from Equation (6.2):

$$\mathcal{L}_{\hat{\alpha}}(f) = \mathbb{E}_{\boldsymbol{x}\sim\hat{\alpha}}\big[a_f(\boldsymbol{x})\big] - \mathbb{E}_{\boldsymbol{y}\sim\hat{\beta}}\big[b_f(\boldsymbol{y})\big] = \big\langle \rho_1, a_f\big\rangle_{L^2(\hat{\gamma})} - \big\langle \rho_2, b_f\big\rangle_{L^2(\hat{\gamma})}, \tag{D.99}$$

hence by composition:

$$\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f) = \rho_1 \cdot \big(a' \circ f\big) - \rho_2 \cdot \big(b' \circ f\big) = \rho_1 a'_f - \rho_2 b'_f. \tag{D.100}$$

□

D.1.6.2. LSGAN

Proposition 5 (LSGAN discriminator)**.** Under Assumptions 1 and 2, the solutions of Equation (6.9) for $a = -(\mathrm{id}+1)^2$ and $b = -(\mathrm{id}-1)^2$ are the functions defined for all $t \in \mathbb{R}_+$ as:

$$f_t = \exp\big(-4t\mathcal{T}_{k,\hat{\gamma}}\big)(f_0 - \rho) + \rho = f_0 + \varphi_t\big(\mathcal{T}_{k,\hat{\gamma}}\big)(f_0 - \rho), \tag{D.101}$$

where:

$$\varphi_t \colon x \mapsto \mathrm{e}^{-4tx} - 1. \tag{D.102}$$

Proof. Assumptions 1 and 2 are already assumed and Assumption 3 holds for the given a and b in LSGAN. Thus, Theorem 1 applies, and there exists a unique solution $t \mapsto f_t$ to Equation (6.9) over $\mathbb{R}_+$ in $L^2(\Omega)$ for a given initial condition f_0. Therefore, there remains to prove that, for a given initial condition f_0,

$$g \colon t \mapsto g_t = f_0 + \varphi_t\big(\mathcal{T}_{k,\hat{\gamma}}\big)(f_0 - \rho) \tag{D.103}$$

is a solution to Equation (6.9) with $g_0 = f_0$ and $g_t \in L^2(\Omega)$ for all $t \in \mathbb{R}_+$.

Let us first express the gradient of $\mathcal{L}_{\hat{\alpha}}$. We have from Lemma 10, with $a_f = -(f+1)^2$ and $b_f = -(f-1)^2$:

$$\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f) = \rho_1 a'_f - \rho_2 b'_f = -2\rho_1(f+1) - 2\rho_2(f-1) = 4\rho - 4f. \tag{D.104}$$

So Equation (6.9) equates to:

$$\partial_t f_t = 4\mathcal{T}_{k,\hat{\gamma}}(\rho - f_t). \tag{D.105}$$

Now let us prove that g_t is a solution to Equation (D.105). We have:

$$\partial_t g_t = -4\Big(\mathcal{T}_{k,\hat{\gamma}} \circ \exp\big(-4t\mathcal{T}_{k,\hat{\gamma}}\big)\Big)(f_0 - \rho) = -4\Big(\mathcal{T}_{k,\hat{\gamma}} \circ \exp\big(-4t\mathcal{T}_{k,\hat{\gamma}}\big)\Big)(f_0 - \rho). \tag{D.106}$$

Restricted to $\operatorname{supp}\hat{\gamma}$, we can write from Equation (D.103):

$$g_t = f_0 + \left(\exp\left(-4t\,\mathcal{T}_{k,\hat{\gamma}}\big|_{\operatorname{supp}\hat{\gamma}}\right) - \mathrm{id}_{L^2(\hat{\gamma})}\right)(f_0 - \rho), \tag{D.107}$$

and plugging this in Equation (D.106):

$$\partial_t g_t = -4\mathcal{T}_{k,\hat{\gamma}}(g_t - \rho), \tag{D.108}$$

where we retrieve the differential equation of Equation (D.105). Therefore, g_t is a solution to Equation (D.105).

It is clear that $g_0 = f_0$. Moreover, $\mathcal{T}_{k,\hat{\gamma}}$ being decomposable in a finite orthonormal basis of elements of operators over $L^2(\Omega)$, its exponential has values in $L^2(\Omega)$ as well, making g_t belong to $L^2(\Omega)$ for all t. With this, the proof is complete. □

D.1.6.3. IPMs

Proposition 4 (IPM discriminator)**.** Under Assumptions 1 and 2, the solutions of Equation (6.9) for $a = b = \mathrm{id}$ are the functions of the form $f_t = f_0 + tf^*_{\hat{\alpha}}$, where $f^*_{\hat{\alpha}}$ is the unnormalized MMD witness function, yielding:

$$f^*_{\hat{\alpha}} = \mathbb{E}_{\boldsymbol{x}\sim\hat{\alpha}}\big[k(\boldsymbol{x},\cdot)\big] - \mathbb{E}_{\boldsymbol{y}\sim\hat{\beta}}\big[k(\boldsymbol{y},\cdot)\big], \quad \mathcal{L}_{\hat{\alpha}}(f_t) = \mathcal{L}_{\hat{\alpha}}(f_0) + t\cdot \mathrm{MMD}^2_k\Big(\hat{\alpha},\hat{\beta}\Big). \tag{D.109}$$

Proof. Assumptions 1 and 2 are already assumed and Assumption 3 holds for the given a and b of the IPM loss. Thus, Theorem 1 applies, and there exists a unique solution $t \mapsto f_t$ to Equation (6.9) over $\mathbb{R}_+$ in $L^2(\Omega)$ for a given initial condition f_0. Therefore, in order to find the solution of Equation (6.9), there remains to prove that, for a given initial condition f_0,

$$g\colon t \mapsto g_t = f_0 + tf^*_{\hat{\alpha}} \tag{D.110}$$

is a solution to Equation (6.9) with $g_0 = f_0$ and $g_t \in L^2(\Omega)$ for all $t \in \mathbb{R}_+$.

Let us first express the gradient of $\mathcal{L}_{\hat{\alpha}}$. We have from Lemma 10, with $a_f = b_f = f$:

$$\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f) = \rho_1 a'_f - \rho_2 b'_f = -2\rho. \tag{D.111}$$

So Equation (6.9) equates to:

$$\partial_t f_t = -2\mathcal{T}_{k,\hat{\gamma}}(\rho) = 2\int_{\boldsymbol{x}} k(\cdot,\boldsymbol{x})\rho(\boldsymbol{x})\,\mathrm{d}\hat{\gamma}(\boldsymbol{x}) = \int_{\boldsymbol{x}} k(\cdot,\boldsymbol{x})\,\mathrm{d}\hat{\alpha}(\boldsymbol{x}) - \int_{\boldsymbol{y}} k(\cdot,\boldsymbol{y})\,\mathrm{d}\hat{\beta}(\boldsymbol{y}), \tag{D.112}$$

by definition of ρ (see Equation (D.96)), yielding:

$$\partial_t f_t = f^*_{\hat{\alpha}}. \tag{D.113}$$

Clearly, $t \mapsto g_t = f_0 + tf^*_{\hat{\alpha}}$ is a solution of the latter equation, $g_0 = f_0$ and $g_t \in L^2(\Omega)$ given that $\operatorname{supp}\hat{\gamma}$ is finite and $k \in L^2\big(\Omega^2\big)$ by assumption. The set of solutions for the IPM loss is thus characterized.

Finally, let us compute $\mathcal{L}_{\hat{\alpha}}(f_t)$. By linearity of $\mathcal{L}_{\hat{\alpha}}$ for $a = b = \mathrm{id}$:

$$\mathcal{L}_{\hat{\alpha}}(f_t) = \mathcal{L}_{\hat{\alpha}}(f_0) + t \cdot \mathcal{L}_{\hat{\alpha}}(f^*_{\hat{\alpha}}) = \mathcal{L}_{\hat{\alpha}}(f_0) + t \cdot \mathcal{L}_{\hat{\alpha}}\big(\mathcal{T}_{k,\hat{\gamma}}(-2\rho)\big). \tag{D.114}$$

But, from Equation (D.99), $\mathcal{L}_{\hat{\alpha}}(f) = \langle -2\rho, f\rangle_{L^2(\hat{\gamma})}$, hence:

$$\mathcal{L}_{\hat{\alpha}}(f_t) = \mathcal{L}_{\hat{\alpha}}(f_0) + t \cdot \big\langle -2\rho, \mathcal{T}_{k,\hat{\gamma}}(-2\rho)\big\rangle_{L^2(\hat{\gamma})} = \mathcal{L}_{\hat{\alpha}}(f_0) + t \cdot \big\|\mathcal{T}_{k,\hat{\gamma}}(-2\rho)\big\|^2_{\mathcal{H}^{\hat{\gamma}}_k}. \tag{D.115}$$

By noticing that $\mathcal{T}_{k,\hat{\gamma}}(-2\rho) = f^*_{\hat{\alpha}}$ and that $\big\|f^*_{\hat{\alpha}}\big\|_{\mathcal{H}^{\hat{\gamma}}_k} = \mathrm{MMD}_k\Big(\hat{\alpha}, \hat{\beta}\Big)$ since $f^*_{\hat{\alpha}}$ is the unnormalized MMD witness function, the expression of $\mathcal{L}_{\hat{\alpha}}(f_t)$ in the proposition is obtained. □

D.1.6.4. Vanilla GAN

Unfortunately, finding the solutions to Equation (6.9) in the case of the original GAN formulation, i.e. $a = \log(1-\sigma)$ and $b = -\log\sigma$, remains to the extent of our knowledge an open problem. We provide in the rest of this section some leads that might prove useful for more advanced analyses.

Let us first determine the expression of Equation (6.9) for vanilla GAN.

Lemma 11. *For $a = \log(1-\sigma)$ and $b = -\log\sigma$, Equation (6.9) equates to:*

$$\partial_t f_t = \mathcal{T}_{k,\hat{\gamma}}\big(\rho_2 - 2\sigma(f)\big). \tag{D.116}$$

Proof. We have from Lemma 10, with $a_f = b_f = f$:

$$\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f) = \rho_1 a'_f - \rho_2 b'_f = -\rho_1 \frac{\sigma'(f)}{1-\sigma(f)} + \rho_2 \frac{\sigma'(f)}{\sigma(f)}. \tag{D.117}$$

By noticing that $\sigma'(f) = \sigma(f)\big(1-\sigma(f)\big)$, we obtain:

$$\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f) = \rho_1 a'_f - \rho_2 b'_f = -\rho_1\sigma(f) + \rho_2\big(1-\sigma(f)\big) = \rho_2 - 2\sigma(f). \tag{D.118}$$

By plugging the latter expression in Equation (6.9), the desired result is achieved. □

Note that Assumption 3 holds for these choices of a and b. Therefore, under Assumptions 1 and 2, there exists a unique solution to Equation (D.116) in $\mathbb{R}_+ \to L^2(\Omega)$ with a given initialization f_0.

Let us first study Equation (D.116) in the simplified case of a one-dimensional ordinary differential equation.

Proposition 8. *Let $r \in \{0, 2\}$ and $\lambda \in \mathbb{R}$. The set of differentiable solutions over $\mathbb{R}$ to this ordinary differential equation:*

$$\frac{\mathrm{d}y_t}{\mathrm{d}t} = \lambda\big(r - 2\sigma(y_t)\big) \tag{D.119}$$

is the following set:

$$S = \left\{ y : t \mapsto (1-r)\left(W\left(\mathrm{e}^{2\lambda t + C}\right) - 2\lambda t - C\right) \;\middle|\; C \in \mathbb{R} \right\}, \tag{D.120}$$

where W the is principal branch of the Lambert W function (Corless, Gonnet, et al., 1996).

Proof. The theorem of Cauchy-Lipschitz ensures that there exists a unique global solution to Equation (D.119) for a given initial condition $y_0 \in \mathbb{R}$. Therefore, we only need to show that all elements of S are solutions of Equation (D.119) and that they can cover any initial condition.

Let us first prove that $y : t \mapsto (1-r)\left(W\left(\mathrm{e}^{2\lambda t + C}\right) - 2\lambda t - C\right)$ is a solution of Equation (D.119). Let us express the derivative of y:

$$\frac{1}{1-r}\frac{\mathrm{d}y_t}{\mathrm{d}t} = 2\lambda\left(\mathrm{e}^{2\lambda t + C} W'\left(\mathrm{e}^{2\lambda t + C}\right) - 1\right). \tag{D.121}$$

$W'(z) = \frac{W(z)}{z\left(1+W(z)\right)}$, so:

$$\frac{1}{1-r}\frac{\mathrm{d}y_t}{\mathrm{d}t} = 2\lambda\left(\frac{W\left(\mathrm{e}^{2\lambda t + C}\right)}{1 + W\left(\mathrm{e}^{2\lambda t + C}\right)} - 1\right) = -\frac{2\lambda}{1 + W\left(\mathrm{e}^{2\lambda t + C}\right)}. \tag{D.122}$$

Moreover, $W(z) = z\mathrm{e}^{-W(z)}$, and with $r - 1 \in \{1, -1\}$:

$$\frac{1}{1-r}\frac{\mathrm{d}y_t}{\mathrm{d}t} = -\frac{2\lambda}{1 + \mathrm{e}^{2\lambda t + C}\mathrm{e}^{-W\left(\mathrm{e}^{2\lambda t + C}\right)}} = -\frac{2\lambda}{1 + \mathrm{e}^{(r-1)y_t}}. \tag{D.123}$$

Finally, we notice that, since $r \in \{0, 2\}$:

$$\lambda\big(r - 2\sigma(y_t)\big) = -\frac{2\lambda(1-r)}{1 + \mathrm{e}^{(r-1)y_t}}. \tag{D.124}$$

Therefore:

$$\frac{\mathrm{d}y_t}{\mathrm{d}t} = \lambda\big(r - 2\sigma(y_t)\big) \tag{D.125}$$

and y_t is a solution to Equation (D.119).

Since $y_0 = (1-r)\left(W\left(\mathrm{e}^{C}\right) - C\right)$ and $z \mapsto W(\mathrm{e}^z) - z$ can be proven to be bijective over $\mathbb{R}$, the elements of S can cover any initial condition. With this, the result is proved. □

Suppose that $f_0 = 0$ in Equation (D.116) and that ρ_2 has values in $\{0, 2\}$ – i.e. $\hat{\alpha}$ and $\hat{\beta}$ have disjoint supports (which is the typical case for distributions with finite support). From Proposition 8, a candidate solution would be:

$$f_t = \varphi_t(x)(\rho_2 - 1) = -\varphi_t(x)(\rho), \tag{D.126}$$

where:

$$\varphi_t \colon x \mapsto W\left(\mathrm{e}^{2tx+1}\right) - 2tx - 1, \tag{D.127}$$

since the initial condition $y_0 = 0$ gives the constant value $C = 1$ in Equation (D.120). Note that the Lambert W function of a symmetric linear operator is well-defined, all the more so as we choose the principal branch of the Lambert function in our case; see the work of Corless, Ding, et al. (2007) for more details. Note also that the estimation of $W(\mathrm{e}^z)$ is actually numerically stable using approximations from Iacono and J. P. Boyd (2017).

However, Equation (D.126) cannot be a solution of Equation (D.116). Indeed, one can prove by following essentially the same reasoning as the proof of Proposition 8 that:

$$\partial_t f_t = 2\left(\mathcal{T}_{k,\hat{\gamma}} \circ \left(\psi_t\left(\mathcal{T}_{k,\hat{\gamma}}\right)\right)^{-1}\right)(\rho_2 - 1), \tag{D.128}$$

with:

$$\psi_t \colon x \mapsto 1 + W\left(\mathrm{e}^{2tx+1}\right) > 0. \tag{D.129}$$

However, this does not allow us to obtain Equation (D.116) since in the latter the sigmoid is taken coordinate-wise, where the exponential in Equation (D.128) acts on matrices.

Nonetheless, for t small enough, f_t as defined in Equation (D.128) should approximate the solution of Equation (D.116), since sigmoid is approximately linear around 0 and $f_t \approx 0$ when t is small enough. We find in practice that for reasonable values of t, e.g. $t \leq 5$, the approximate solution of Equation (D.128) is actually close to the numerical solution of Equation (D.116) obtained using an ODE solver. Thus, we provide here a candidate approximate expression for the discriminator in the setting of the original GAN formulation – i.e. for binary classifiers. We leave for future work a more in-depth study of this case.

D.2. Discussions and Remarks

We develop in this section some remarks and explanations referenced in Chapter 6.

D.2.1. From Finite to Infinite-Width Networks

The constancy of the neural tangent kernel during training when the width of the network becomes increasingly large is broadly applicable. As summarized by C. Liu, Libin Zhu, and Belkin (2020), typical neural networks with the building blocks of multilayer perceptrons and convolutional neural networks comply with this property, as long as they end with a linear layer and they do not have any bottleneck – indeed, this constancy needs the minimum internal width to grow unbounded (Arora, Du, W. Hu, et al., 2019). This includes, for example, residual convolutional neural networks (K. He et al., 2016). The requirement of a final linear activation can be circumvented by transferring this activation into the loss function, as we did for the original GAN

formulation in Section 6.3. This makes our framework encompass a wide range of discriminator architectures.

Indeed, many building blocks of state-of-the-art discriminators can be studied in this infinite-width regime with a constant NTK, as highlighted by the exhaustiveness of the Neural Tangents library (Novak et al., 2020). Assumptions about the used activation functions are mild and include many standard activations such as ReLU, sigmoid and tanh. Beyond fully connected linear layers and convolutions, typical operations such as self-attention (Hron et al., 2020), layer normalization and batch normalization (G. Yang, 2020). This variety of networks affected by the constancy of the NTK supports the generality of our approach, as it includes powerful discriminator architectures such as BigGAN (Brock, Donahue, and Simonyan, 2019).

There are nevertheless some limits to this approximation, as we are not aware of works studying the application of the infinite-width regime to some operations such as spectral normalization, and networks in the regime of a constant NTK cannot perform feature learning as they are equivalent to kernel methods (M. Geiger et al., 2020; G. Yang and E. J. Hu, 2021). However, this framework remains general and constitutes the most advanced attempt at theoretically modeling the discriminator's architecture in GANs.

D.2.2. Loss of the Generator and its Gradient

We highlight in this section the importance of taking into account discriminator gradients in the optimization of the generator. Let us focus on an example similar to the one of Arjovsky, Chintala, and Bottou (2017, Example 1) and choose as β a single Dirac centered at $\mathbf{0}$ and as $\alpha_g = \alpha_\theta$ single Dirac centered at $\boldsymbol{x}_\theta = \theta$ (the generator parameters being the coordinates of the generated point). Let us focus for the sake of simplicity on the case of LSGAN since it is a recurring example in this work, but a similar reasoning can be done for other GAN instances.

In the theoretical min-max formulation of GANs considered by Arjovsky, Chintala, and Bottou (2017), the generator is trained to minimize the following quantity:

$$\mathcal{C}_{f^\star_{\alpha_\theta}}(\alpha_\theta) \triangleq \mathbb{E}_{\boldsymbol{x}\sim\alpha_\theta}\left[c_{f^\star_{\alpha_\theta}}(\boldsymbol{x})\right] = f^\star_{\alpha_\theta}(\boldsymbol{x}_\theta)^2, \tag{D.130}$$

where:

$$\begin{aligned} f^\star_{\alpha_\theta} &= \underset{f\in L^2\left(\frac{1}{2}\alpha_\theta+\frac{1}{2}\beta\right)}{\arg\max} \left\{\mathcal{L}_{\alpha_\theta}(f) \triangleq \mathbb{E}_{\boldsymbol{x}\sim\alpha_\theta}\left[a_f(\boldsymbol{x})\right] - \mathbb{E}_{\boldsymbol{y}\sim\beta}\left[b_f(\boldsymbol{y})\right]\right\} \\ &= \underset{f\in L^2\left(\frac{1}{2}\alpha_\theta+\frac{1}{2}\beta\right)}{\arg\min} \left\{\left(f^\star_{\alpha_\theta}(\boldsymbol{x}_\theta)+1\right)^2 + \left(f^\star_{\alpha_\theta}(\mathbf{0})-1\right)^2\right\}. \end{aligned} \tag{D.131}$$

Consequently, $f^\star_{\alpha_\theta}(\mathbf{0}) = 1$ and $f^\star_{\alpha_\theta}(\boldsymbol{x}_\theta) = -1$ when $\boldsymbol{x}_\theta \neq \mathbf{0}$, thus in this case:

$$\mathcal{C}_{f^\star_{\alpha_\theta}}(\alpha_\theta) = 1. \tag{D.132}$$

This constancy of the generator loss would make it impossible to be learned by gradient descent, as pointed out by Arjovsky, Chintala, and Bottou (2017).

However, the setting does not correspond to the actual optimization process used in practice and represented by Equation (6.3). We do have $\nabla_\theta \mathcal{C}_{f^\star_{\alpha_\theta}}(\alpha_\theta) = \mathbf{0}$ when $\boldsymbol{x}_\theta \neq \mathbf{0}$, but the generator never uses this gradient in standard GAN optimization. Indeed, this gradient takes into account the dependency of the optimal discriminator $f^\star_{\alpha_\theta}$ in the generator parameters, since the optimal discriminator depends on the generated distribution. Yet, in practice and with few exceptions such as Unrolled GANs (Metz et al., 2017) and as done in Equation (6.3), this dependency is ignored when computing the gradient of the generator, because of the alternating optimization setting – where the discriminator is trained in-between generator's updates. Therefore, despite being constant on the training data, this loss can yield non-zero gradients to the generator. However, this requires the gradient of $f^\star_{\alpha_\theta}$ to be defined, which is the issue addressed in Section 6.3.2.

D.2.3. Differentiability of the Bias-Free ReLU Kernel

Remark 1 contradicts the results of Bietti and Mairal (2019) on the regularity of the NTK of a bias-free ReLU MLP with one hidden layer, which can be expressed as follows (up to a constant scaling the matrix multiplication in linear layers):

$$k(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x}\|\|\boldsymbol{y}\|\kappa\left(\frac{\langle \boldsymbol{x}, \boldsymbol{y}\rangle}{\|\boldsymbol{x}\|\|\boldsymbol{y}\|}\right), \tag{D.133}$$

where:

$$\begin{aligned} \kappa\colon [0,1] &\to \mathbb{R} \\ u &\mapsto \frac{2}{\pi}u(\pi - \arccos u) + \frac{1}{\pi}\sqrt{1-u^2}\,. \end{aligned} \tag{D.134}$$

More particularly, Bietti and Mairal (2019, Proposition 3) claim that $k(\cdot, \boldsymbol{y})$ is not Lipschitz around $\boldsymbol{y}$ for all $\boldsymbol{y}$ in the unit sphere. By following their proof, it amounts to prove that $k(\cdot, \boldsymbol{y})$ is not Lipschitz around $\boldsymbol{y}$ for all $\boldsymbol{y}$ in any centered sphere. We highlight that this also contradicts empirical evidence, as we did observe the Lipschitzness of such NTK in practice using the Neural Tangents library (Novak et al., 2020).

We believe that the mistake in the proof of Bietti and Mairal (2019) lies in the confusion between functions κ and k_0, with:

$$k_0\colon \boldsymbol{x}, \boldsymbol{y} \mapsto \kappa\left(\frac{\langle \boldsymbol{x}, \boldsymbol{y}\rangle}{\|\boldsymbol{x}\|\|\boldsymbol{y}\|}\right). \tag{D.135}$$

Their proof relies on the fact that κ is non-Lipschitz in the neighborhood of $u = 1$. However, this does not imply that k_0 is not Lipschitz or not derivable, because κ and k_0 have different geometries. We can prove that it is actually at least locally Lipschitz.

Indeed, let us compute the following derivative for $\boldsymbol{x} \neq \boldsymbol{y} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$:

$$\frac{\partial k_0(\boldsymbol{x}, \boldsymbol{y})}{\partial \boldsymbol{x}} = \frac{\boldsymbol{y}\|\boldsymbol{x}\| - \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}\langle \boldsymbol{x}, \boldsymbol{y}\rangle}{\|\boldsymbol{x}\|^2\|\boldsymbol{y}\|}\kappa'(u) = \frac{1}{\|\boldsymbol{x}\|\|\boldsymbol{y}\|}\left(\boldsymbol{y} - \langle \boldsymbol{x}, \boldsymbol{y}\rangle\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|^2}\right)\kappa'(u), \tag{D.136}$$

where $u = \frac{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}{\|\boldsymbol{x}\|\|\boldsymbol{y}\|}$ and:

$$\pi \cdot \kappa'(u) = \frac{u}{\sqrt{1-u^2}} + 2(\pi - \arccos u). \tag{D.137}$$

Note that $\kappa'(u) \sim_{u \to 1^-} \frac{\pi u}{\sqrt{1-u^2}} \sim_{u \to 1^-} \frac{\pi}{\sqrt{2}\sqrt{1-u}}$. Therefore:

$$\begin{aligned} \frac{\pi}{\sqrt{2}} \cdot \frac{\partial k_0(\boldsymbol{x}, \boldsymbol{y})}{\partial \boldsymbol{x}} &\sim_{\boldsymbol{x} \to \boldsymbol{y}} \frac{1}{\|\boldsymbol{y}\|^2} \left(\boldsymbol{y} - \langle \boldsymbol{x}, \boldsymbol{y} \rangle \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|^2} \right) \frac{\sqrt{\|\boldsymbol{x}\|\|\boldsymbol{y}\|}}{\sqrt{\|\boldsymbol{x}\|\|\boldsymbol{y}\| - \langle \boldsymbol{x}, \boldsymbol{y} \rangle}} \\ &\sim_{\boldsymbol{x} \to \boldsymbol{y}} \frac{\|\boldsymbol{x}\|^2 \boldsymbol{y} - \langle \boldsymbol{x}, \boldsymbol{y} \rangle \boldsymbol{x}}{\|\boldsymbol{y}\|^3 \sqrt{\|\boldsymbol{x}\|\|\boldsymbol{y}\| - \langle \boldsymbol{x}, \boldsymbol{y} \rangle}} \\ &\sim_{\boldsymbol{x} \to \boldsymbol{y}} \frac{\|\boldsymbol{y}\|^2 - \langle \boldsymbol{x}, \boldsymbol{y} \rangle}{\|\boldsymbol{y}\|^3 \sqrt{\|\boldsymbol{y}\|^2 - \langle \boldsymbol{x}, \boldsymbol{y} \rangle}} \boldsymbol{y} \xrightarrow[\boldsymbol{x} \to \boldsymbol{y}]{} \boldsymbol{0}, \end{aligned} \tag{D.138}$$

which proves that k_0 is actually Lipschitz around points $(\boldsymbol{y}, \boldsymbol{y})$, as well as differentiable, and confirms our remark.

D.2.4. Integral Operator and Instance Noise

Instance noise (C. K. Sønderby, Caballero, et al., 2017) consists in adding random Gaussian noise to the input and target samples. This amounts to convolving the data distributions with a Gaussian density, which will have the effect of smoothing the discriminator. In the following, for the case of IPM losses, we link instance noise with our framework, showing that smoothing of the data distributions already occurs via the NTK kernel, stemming from the fact that the discriminator is a neural network trained with gradient descent.

More specifically, it can be shown that if k is an RBF kernel, the optimal discriminators in both case are the same. This is based on the fact that the density of a convolution of an empirical measure $\hat{\mu} = \frac{1}{N} \sum_i \delta_{\boldsymbol{x}_i}$, where $\delta_{\boldsymbol{z}}$ is the Dirac distribution centered on $\boldsymbol{z}$, and a Gaussian density $\tilde{k}$ with associated RBF kernel k can be written as $\tilde{k} * \hat{\mu} = \frac{1}{N} \sum_i k(\boldsymbol{x}_i, \cdot)$.

Let us consider the following regularized discriminator optimization problem in $L^2(\mathbb{R})$ smoothed from $L^2(\Omega)$ with instance noise, i.e. convolving $\hat{\alpha}$ and $\hat{\beta}$ with $\tilde{k}$.

$$\sup_{f \in L^2(\mathbb{R})} \left\{ \mathbb{E}_{\boldsymbol{x} \sim \tilde{k} * \hat{\alpha}} \big[f(\boldsymbol{x}) \big] - \mathbb{E}_{\boldsymbol{y} \sim \tilde{k} * \hat{\beta}} \big[f(\boldsymbol{y}) \big] - \lambda \|f\|_{L^2}^2 \right\} \tag{D.139}$$

The optimum f^{IN} can be found by taking the gradient:

$$\nabla_f \left(\mathcal{L}_{\hat{\alpha}}^{\tilde{k}} \left(f^{\mathrm{IN}} \right) - \lambda \left\| f^{\mathrm{IN}} \right\|_{L^2}^2 \right) = 0 \qquad \Leftrightarrow \qquad f^{\mathrm{IN}} = \frac{1}{2\lambda} \Big(\tilde{k} * \hat{\alpha} - \tilde{k} * \hat{\beta} \Big). \tag{D.140}$$

If we now study the resolution of the optimization problem in $\mathcal{H}_k^{\hat{\gamma}}$ as in Section 6.5.1 with $f_0 = 0$, we find the following discriminator:

$$f_t = t\Big(\mathbb{E}_{\boldsymbol{x} \sim \hat{\alpha}} \big[k(\boldsymbol{x}, \cdot) \big] - \mathbb{E}_{\boldsymbol{y} \sim \hat{\beta}} \big[k(\boldsymbol{y}, \cdot) \big] \Big) = t\Big(\tilde{k} * \hat{\alpha} - \tilde{k} * \hat{\beta} \Big). \tag{D.141}$$

Therefore, we have that $f^{\text{IN}} \propto f_t$, i.e. instance noise and regularization by neural networks obtain the same smoothed solution.

This analysis was done using the example of an RBF kernel, but it also holds for stationary kernels, i.e. $k(\boldsymbol{x}, \boldsymbol{y}) = \tilde{k}(\boldsymbol{x} - \boldsymbol{y})$, which can be used to convolve measures. We remind that this is relevant, given that NTKs are stationary over spheres (Jacot, Gabriel, and Hongler, 2018; G. Yang and Salman, 2019), around where data can be concentrated in high dimensions.

D.2.5. Positive Definite NTKs

Optimality results in the theory of NTKs usually rely on the assumption that the considered NTK k is positive definite over the training dataset $\hat{\gamma}$ (Jacot, Gabriel, and Hongler, 2018; Yaoyu Zhang et al., 2020). This property offers several theoretical advantages.

Indeed, this gives sufficient representational power to its RKHS to include the optimal solution over $\hat{\gamma}$. Moreover, this positive definiteness property equates for finite datasets to the invertibility of the mapping

$$\begin{aligned} \mathcal{T}_{k,\hat{\gamma}}\big|_{\operatorname{supp}\hat{\gamma}} \colon L^2(\hat{\gamma}) &\to L^2(\hat{\gamma}) \\ h &\mapsto \mathcal{T}_{k,\hat{\gamma}}(h)\big|_{\operatorname{supp}\hat{\gamma}} \end{aligned}, \tag{D.142}$$

that can be seen as a multiplication by the invertible Gram matrix of k over $\hat{\gamma}$. From this, one can retrieve the expression of $f \in \mathcal{H}_k^{\hat{\gamma}}$ from its restriction $f|_{\operatorname{supp}\hat{\gamma}}$ to $\operatorname{supp}\hat{\gamma}$ in the following way:

$$f = \mathcal{T}_{k,\hat{\gamma}} \circ \mathcal{T}_{k,\hat{\gamma}}\big|^{-1}_{\operatorname{supp}\hat{\gamma}} \Big(f|_{\operatorname{supp}\hat{\gamma}} \Big), \tag{D.143}$$

as shown in Lemma 9. Finally, as shown by Jacot, Gabriel, and Hongler (2018) and in Appendix D.1.5, this makes the discriminator loss function strictly increase during training.

One may wonder whether this assumption is reasonable for NTKs. Jacot, Gabriel, and Hongler (2018) proved that it indeed holds for NTKs of non-shallow MLPs with non-polynomial activations if data is supported on the unit sphere, supported by the fact that the NTK is stationary over the unit sphere. Others, such as Z. Fan and Zhichao Wang (2020), have observed positive definiteness of the NTK subject to specific assumptions on the networks and data. We are not aware of more general results of this kind. However, one may conjecture that, at least for specific kinds of networks, NTKs are positive definite for any training data.

Indeed, besides global convergence results (Allen-Zhu, Yuanzhi Li, and Z. Song, 2019), prior work indicates that MLPs are universal approximators (Hornik, Stinchcombe, and White, 1989; Leshno et al., 1993). This property can be linked in our context to universal kernels (Steinwart, 2001), which are guaranteed to be positive definite over any training data (Sriperumbudur, Fukumizu, and Lanckriet, 2011). Universality is linked to the density of the kernel RKHS in the space of continuous functions. In the case of NTKs, previously cited approximation properties can be interpreted as signs of expressive RKHSs, and thus support the hypothesis of universal NTKs. Furthermore,

beyond positive definiteness, universal kernels are also characteristic (Sriperumbudur, Fukumizu, and Lanckriet, 2011), which is interesting when they are used to compute MMDs, as we do in Section 6.5.1. Note that for the standard case of ReLU MLPs, Ji, Telgarsky, and Xian (2020) showed universal approximation results in the infinite-width regime, and works such as the one of Lin Chen and S. Xu (2021) observed that their RKHS is close to the one of the Laplace kernel, which is positive definite.

Bias-free ReLU NTKs are not characteristic. As already noted by Leshno et al. (1993), the presence of bias is important when it comes to the representational power of MLPs. We can retrieve this observation in our framework. In the case of a ReLU shallow network with one hidden layer and without bias, Bietti and Mairal (2019) determine its associated NTK as follows (up to a constant scaling the matrix multiplication in linear layers):

$$k(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x}\|\|\boldsymbol{y}\|\kappa\left(\frac{\langle \boldsymbol{x}, \boldsymbol{y}\rangle}{\|\boldsymbol{x}\|\|\boldsymbol{y}\|}\right), \tag{D.144}$$

with in particular $k(\boldsymbol{x}, \boldsymbol{0}) = \boldsymbol{0}$ for all $\boldsymbol{x} \in \Omega$; suppose that $\boldsymbol{0} \in \Omega$. This expression of the kernel implies that k is not positive definite for all datasets: take for example $\boldsymbol{x} = \boldsymbol{0}$ and $\boldsymbol{y} \in \Omega \setminus \{\boldsymbol{0}\}$; then the Gram matrix of k has a null row, hence k is not strictly positive definite over $\{\boldsymbol{x}, \boldsymbol{y}\}$. Another consequence is that k is not characteristic. Indeed, take probability distributions $\mu = \delta_{\frac{\boldsymbol{y}}{2}}$ and $\nu = \frac{1}{2}\big(\delta_{\boldsymbol{x}} + \delta_{\boldsymbol{y}}\big)$ with $\delta_{\boldsymbol{z}}$ being the Dirac distribution centered on $\boldsymbol{z} \in \Omega$, and where $\boldsymbol{x} = \boldsymbol{0}$ and $\boldsymbol{y} \in \Omega \setminus \{\boldsymbol{0}\}$. Then:

$$\mathbb{E}_{\boldsymbol{z}\sim\mu} k(\boldsymbol{z}, \cdot) = k\left(\frac{1}{2}\boldsymbol{y}, \cdot\right) = \frac{1}{2}k(\boldsymbol{y}, \cdot) = \frac{1}{2}\big(k(\boldsymbol{y}, \cdot) + k(\boldsymbol{x}, \cdot)\big) = \mathbb{E}_{\boldsymbol{z}\sim\nu} k(\boldsymbol{z}, \cdot), \tag{D.145}$$

i.e. kernel embeddings of μ and $\nu \neq \mu$ are identical, making k not characteristic by definition.

D.3. Experimental Details

We detail in this section the experimental parameters needed to reproduce our experiments.

All two-dimensional experiments require only a few minutes of computations on a single GPU. Experiments on MNIST were run using simultaneously four GPUs for parallel computations, for at most a couple of hours.

D.3.1. Datasets

8 **Gaussians.** The target distribution is composed of 8 Gaussians with their means being evenly distributed on the centered sphere of radius 5, and each with a standard deviation of 0.5. The input fake distribution is drawn at initialization from a standard normal distribution $\mathcal{N}(0, 1)$. We sample in our experiments 500 points from each distribution at each run to build $\hat{\alpha}$ and $\hat{\beta}$.

AB and Density. These two datasets are taken from the Geomloss library examples (Feydy et al., 2019).[1] To sample a point from a distribution based on these greyscale images files, we sample a pixel (considered to lie in $[-1, 1]^2$) in the image from a distribution where each pixel probability is proportional to the darkness of this pixel, and then apply a Gaussian noise centered at the chosen pixel coordinates with a standard deviation equal to the inverse of the image size. We sample in our experiments 500 points from each distribution at each run to build $\hat{\alpha}$ and $\hat{\beta}$.

MNIST and CelebA. We preprocess each MNIST image by extending it from 28×28 frames to 32×32 frames (by padding it with black pixels). CelebA images are downsampled from a size of 178×218 to 32×39 and then center-cropped to 32×32.

For both datasets, we normalize pixels in the $[-1, 1]$ range. For our experiments, we consider a subset of 1024 elements of each dataset, which are randomly sampled for each run.

D.3.2. Parameters

Sinkhorn divergence. The Sinkhorn divergence is computed using the Geomloss library (Feydy et al., 2019), with a blur parameter of 0.001 and a scaling of 0.95, making it close to the Wasserstein $\mathcal{W}_2$ distance.

Sigmoid activation. We introduce in Section 6.6 a sigmoid-like activation $\tilde{\sigma}$, that we abbreviate to sigmoid in this experimental study for readability purposes. We choose $\tilde{\sigma}$ instead of the actual sigmoid σ for computational reasons, since $\tilde{\sigma}$, contrary to σ, allows for analytic computations of NTKs in the Neural Tangents library (Novak et al., 2020). $\tilde{\sigma}$ is defined in the latter using the error function erf scaled in order to minimize a squared loss with respect to σ over $[-5, 5]$, with the following expression:

$$\tilde{\sigma} \colon x \mapsto \frac{1}{2}\left(\operatorname{erf}\left(\frac{x}{2.402\,056\,353\,171\,979\,6}\right) + 1\right). \tag{D.146}$$

RBF kernel. The RBF kernel used in our experiments is the following:

$$k(\boldsymbol{x}, \boldsymbol{y}) = \mathrm{e}^{\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{2n}}, \tag{D.147}$$

where n is the dimension of $\boldsymbol{x}$ and $\boldsymbol{y}$, i.e. the dimension of the data.

Architecture. We used for the neural networks of our experiments the standard NTK paramaterization (Jacot, Gabriel, and Hongler, 2018), with a scaling factor of 1 for matrix multiplications and, when bias in enabled, a multiplicative constant of 1 for biases (except for sigmoid where this bias factor is lowered to 0.2 to avoid saturating

[1] They can be downloaded at `https://github.com/jeanfeydy/geomloss/tree/master/geomloss/examples/optimal_transport/data`: AB corresponds to files `A.png` (source) and `B.png` (target), and Density corresponds to files `density_a.png` (source) and `density_a.png` (target).

the sigmoid, and for CelebA where it is equal to 4). All considered networks are composed of 3 hidden layers and end with a linear layer. In the finite-width case, the width of these hidden layers is 128. We additionally use antisymmetrical initialization (Yaoyu Zhang et al., 2020), except for the finite-width LSGAN loss.

Discriminator optimization. Discriminators in the finite-width regime are trained using full-batch gradient descent without momentum, with one step per update to the distributions and the following learning rates ε:

- for the IPM loss: $\varepsilon = 0.01$;
- for the IPM loss with reset and LSGAN: $\varepsilon = 0.1$.

In the infinite-width limit, we use the analytic expression derived in Section 6.5 with training time $\tau = 1$ (except for MNIST and CelebA where $\tau = 1000$).

Point cloud descent. The multiplicative constant η over the gradient applied to each data point for two-dimensional problems is chosen as follows:

- for the IPM loss in the infinite-width regime: $\eta = 1000$;
- for the IPM loss in the finite-width regime: $\eta = 100$;
- for the IPM loss in the finite-width regime and discriminator reset: $\eta = 1000$;
- for LSGAN in the infinite-width regime: $\eta = 1000$;
- for LSGAN in the finite-width regime: $\eta = 1$.

We multiply η by 1000 when using sigmoid activations, because of the low magnitude of the gradients it provides. We choose for MNIST $\eta = 100$.

Training is performed for the following number of iterations:

- for 8 Gaussians: 20 000;
- for Density and AB: 10 000;
- for MNIST: 50 000.

Bibliography

Abadi, Martín, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng (Nov. 2016). "TensorFlow: A System for Large-Scale Machine Learning". In: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. Savannah, Georgia, United States of America: USENIX Association, pp. 265–283 (cit. on p. 5).

Achille, Alessandro and Stefano Soatto (2018). "Emergence of Invariance and Disentanglement in Deep Representations". In: *Journal of Machine Learning Research* 19.50, pp. 1–34 (cit. on p. 197).

Adler, Robert J. (1981). *The Geometry Of Random Fields*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics (cit. on p. 219).

– (1990). "An Introduction to Continuity, Extrema, and Related Topics for General Gaussian Processes". In: *Lecture Notes-Monograph Series* 12, pp. i–155 (cit. on p. 220).

Aksan, Emre and Otmar Hilliges (2019). "STCN: Stochastic Temporal Convolutional Networks". In: *International Conference on Learning Representations* (cit. on p. 19).

Alemi, Alexander A., Ben Poole, Ian Fischer, Joshua Dillon, Rif A. Saurous, and Kevin Murphy (July 2018). "Fixing a Broken ELBO". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 159–168 (cit. on p. 26).

Alemohammad, Sina, Zichao Wang, Randall Balestriero, and Richard G. Baraniuk (2021). "The Recurrent Neural Tangent Kernel". In: *International Conference on Learning Representations* (cit. on p. 105).

Allen-Zhu, Zeyuan, Yuanzhi Li, and Zhao Song (June 2019). "A Convergence Theory for Deep Learning via Over-Parameterization". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 242–252 (cit. on p. 239).

Almeida, Luis B. (1990). "A Learning Rule for Asynchronous Perceptrons with Feedback in a Combinatorial Environment". In: *Artificial Neural Networks: Concept Learning*. IEEE Press, pp. 102–111 (cit. on p. 16).

Ambrosio, Luigi and Gianluca Crippa (2014). "Continuity equations and ODE flows with non-smooth velocity". In: *Proceedings of the Royal Society of Edinburgh: Section A Mathematics* 144.6, pp. 1191–1244 (cit. on p. 227).

Ambrosio, Luigi, Nicola Gigli, and Giuseppe Savaré (2008). *Gradient Flows. In Metric Spaces and in the Space of Probability Measures*. Lectures in Mathematics ETH Zürich. Basel, Switzerland: Birkhäuser Basel (cit. on pp. 111, 114).

Antoniou, Antreas, Amos Storkey, and Harrison Edwards (2017). "Data Augmentation Generative Adversarial Networks". In: arXiv: 1711.04340 (cit. on p. 6).

Arbel, Michael, Anna Korba, Adil Salim, and Arthur Gretton (2019). "Maximum Mean Discrepancy Gradient Flow". In: *Advances in Neural Information Processing Systems*. Ed. by Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily Fox, and Roman Garnett. Vol. 32. Curran Associates, Inc., pp. 6484–6494 (cit. on pp. 111, 113, 114, 116).

Arjovsky, Martin and Léon Bottou (2017). "Towards Principled Methods for Training Generative Adversarial Networks". In: *International Conference on Learning Representations* (cit. on pp. 104, 106, 110).

Arjovsky, Martin, Soumith Chintala, and Léon Bottou (Aug. 2017). "Wasserstein Generative Adversarial 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. PMLR, pp. 214–223 (cit. on pp. 27, 105, 106, 113, 236).

Arora, Sanjeev, Simon S. Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang (2019). "On Exact Computation with an Infinitely Wide Neural Net". In: *Advances in Neural Information Processing Systems*. Ed. by Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily Fox, and Roman Garnett. Vol. 32. Curran Associates, Inc., pp. 8141–8150 (cit. on pp. 104, 225, 235).

Arora, Sanjeev, Simon S. Du, Zhiyuan Li, Ruslan Salakhutdinov, Ruosong Wang, and Dingli Yu (2020). "Harnessing the Power of Infinitely Wide Deep Nets on Small-data Tasks". In: *International Conference on Learning Representations* (cit. on pp. 104, 119).

Arora, Sanjeev, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang (Aug. 2017). "Generalization and Equilibrium in Generative Adversarial Nets (GANs)". 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. PMLR, pp. 224–232 (cit. on p. 106).

Aubry, Mathieu, Daniel Maturana, Alexei A. Efros, Bryan C. Russell, and Josef Sivic (June 2014). "Seeing 3D Chairs: Exemplar Part-based 2D-3D Alignment using a Large Dataset of CAD Models". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3762–3769 (cit. on pp. 96, 200).

Ayed, Ibrahim, Emmanuel de Bézenac, Arthur Pajot, and Patrick Gallinari (May 2020). "Learning the Spatio-Temporal Dynamics of Physical Processes from Partial Observations". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3232–3236 (cit. on pp. 17, 22, 83, 207).

Babaeizadeh, Mohammad, Chelsea Finn, Dumitru Erhan, Roy Campbell, and Sergey Levine (2018). "Stochastic Variational Video Prediction". In: *International Conference on Learning Representations* (cit. on pp. 31, 63–65, 72).

Bachman, Philip, R Devon Hjelm, and William Buchwalter (2019). "Learning Representations by Maximizing Mutual Information Across Views". In: *Advances in*

Neural Information Processing Systems. Ed. by Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily Fox, and Roman Garnett. Vol. 32. Curran Associates, Inc., pp. 15535–15545 (cit. on p. 6).

Bagnall, Anthony, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, Paul Southam, and Eamonn Keogh (2018). "The UEA multivariate time series classification archive, 2018". In: arXiv: 1811.00075 (cit. on pp. 39, 52, 53, 145, 162).

Bagnall, Anthony, Jason Lines, Aaron Bostrom, James Large, and Eamonn Keogh (2017). "The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances". In: *Data Mining and Knowledge Discovery* 31, pp. 606–660 (cit. on pp. 47, 57).

Bai, Shaojie, J. Zico Kolter, and Vladlen Koltun (2018). "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling". In: arXiv: 1803.01271 (cit. on pp. 43, 44).

Bai, Yu, Tengyu Ma, and Andrej Risteski (2019). "Approximability of Discriminators Implies Diversity in GANs". In: *International Conference on Learning Representations* (cit. on pp. 19, 104).

Balaji, Yogesh, Mohammadmahdi Sajedi, Neha Mukund Kalibhat, Mucong Ding, Dominik Stöger, Mahdi Soltanolkotabi, and Soheil Feizi (2021). "Understanding Overparameterization in Generative Adversarial Networks". In: *International Conference on Learning Representations* (cit. on p. 104).

Balduzzi, David, Sebastien Racanière, James Martens, Jakob Foerster, Karl Tuyls, and Thore Graepel (July 2018). "The Mechanics of n-Player Differentiable Games". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 354–363 (cit. on p. 18).

Balntas, Vassileios, Edgar Riba, Daniel Ponsa, and Krystian Mikolajczyk (Sept. 2016). "Learning local feature descriptors with triplets and shallow convolutional neural networks". In: *Proceedings of the British Machine Vision Conference (BMVC)*. Ed. by Richard C. Wilson, Edwin R. Hancock, and William A. P. Smith. BMVA Press, pp. 119.1–119.11 (cit. on pp. 20, 21).

Banach, Stefan (1922). "Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales". In: *Fundamenta Mathematicae* 3.1, pp. 133–181 (cit. on p. 215).

Barakat, Anas and Pascal Bianchi (2021). "Convergence and Dynamical Behavior of the ADAM Algorithm for Nonconvex Stochastic Optimization". In: *SIAM Journal on Optimization* 31.1, pp. 244–274 (cit. on p. 18).

Basri, Ronen, Meirav Galun, Amnon Geifman, David Jacobs, Yoni Kasten, and Shira Kritchman (July 2020). "Frequency Bias in Neural Networks for Input of Non-Uniform Density". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 685–694 (cit. on pp. 105, 119).

Bayer, Justin and Christian Osendorfer (2014). "Learning Stochastic Recurrent Networks". In: arXiv: 1411.7610 (cit. on pp. 33, 65).

Behrmann, Jens, Will Grathwohl, Ricky T. Q. Chen, David Duvenaud, and Jörn-Henrik Jacobsen (June 2019). "Invertible Residual Networks". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, United States of America: PMLR, pp. 573–582 (cit. on pp. 29, 70, 198).

Belotto da Silva, André and Maxime Gazeau (2020). "A General System of Differential Equations to Model First-Order Adaptive Algorithms". In: *Journal of Machine Learning Research* 21.129, pp. 1–42 (cit. on p. 18).

Benenti, Sergio (1997). "Intrinsic characterization of the variable separation in the Hamilton-Jacobi equation". In: *Journal of Mathematical Physics* 38.12, pp. 6578–6602 (cit. on p. 85).

Bengio, Yoshua, Aaron Courville, and Pascal Vincent (Aug. 2013). "Representation Learning: A Review and New Perspectives". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8, pp. 1798–1828 (cit. on pp. 21, 22).

Benzoni-Gavage, Sylvie (2010). *Calcul différentiel et équations différentielles. Cours et exercices corrigés*. Sciences Sup. Dunod (cit. on p. 16).

Berndt, Donald J. and James Clifford (1994). "Using Dynamic Time Warping to Find Patterns in Time Series". In: *AAAI Workshop on Knowledge Discovery in Databases*. Seattle, Washington, United States of America: AAAI Press, pp. 359–370 (cit. on p. 47).

Bertasius, Gedas, Heng Wang, and Lorenzo Torresani (July 2021). "Is Space-Time Attention All You Need for Video Understanding?" In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, pp. 813–824 (cit. on p. 20).

Bietti, Alberto and Francis Bach (2021). "Deep Equals Shallow for ReLU Networks in Kernel Regimes". In: *International Conference on Learning Representations* (cit. on p. 104).

Bietti, Alberto and Julien Mairal (2019). "On the Inductive Bias of Neural Tangent Kernels". In: *Advances in Neural Information Processing Systems*. Ed. by Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily Fox, and Roman Garnett. Vol. 32. Curran Associates, Inc., pp. 12893–12904 (cit. on pp. 104, 105, 110, 237, 240).

Bostrom, Aaron and Anthony Bagnall (2015). "Binary Shapelet Transform for Multiclass Time Series Classification". In: *Big Data Analytics and Knowledge Discovery*. Ed. by Sanjay Madria and Takahiro Hara. Cham, Switzerland: Springer International Publishing, pp. 257–269 (cit. on p. 47).

Bottou, Léon, Françoise Fogelman Soulié, Pascal Blanchet, and Jean-Sylvain Liénard (1990). "Speaker-Independent Isolated Digit Recognition: Multilayer Perceptrons vs. Dynamic Time Warping". In: *Neural Networks* 3.4, pp. 453–465 (cit. on p. 19).

Bradbury, James, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang (2018). *JAX: composable transformations of Python+NumPy programs*. Version 0.2.12. URL: http://github.com/google/jax (cit. on p. 114).

Bredin, Hervé (Mar. 2017). "TristouNet: Triplet loss for speaker turn embedding". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5430–5434 (cit. on pp. 21, 40).

Brock, Andrew, Jeff Donahue, and Karen Simonyan (2019). "Large Scale GAN Training for High Fidelity Natural Image Synthesis". In: *International Conference on Learning Representations* (cit. on pp. 28, 113, 138, 236).

Bromley, Jane, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah (1994). "Signature Verification using a "Siamese" Time Delay Neural Network". In: *Advances in Neural Information Processing Systems*. Ed. by Jack D. Cowan, Gerald J. Tesauro, and Joshua Alspector. Vol. 6. Morgan Kaufmann, pp. 737–744 (cit. on p. 20).

Brown, Andrew A. and Michael C. Bartholomew-Biggs (1989). "Some Effective Methods for Unconstrained Optimization Based on the Solution of Systems of Ordinary Differential Equations". In: *Journal of Optimization Theory and Applications* 62, pp. 211–224 (cit. on p. 18).

Brown, Tom, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei (2020). "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. Vol. 33. Curran Associates, Inc., pp. 1877–1901 (cit. on pp. 6, 19).

Brunton, Steven L., Joshua L. Proctor, and J. Nathan Kutz (2016). "Discovering governing equations from data by sparse identification of nonlinear dynamical systems". In: *Proceedings of the National Academy of Sciences* 113.15, pp. 3932–3937 (cit. on p. 84).

Bungartz, Hans-Joachim and Michael Griebel (2004). "Sparse grids". In: *Acta Numerica* 13, pp. 147–269 (cit. on pp. 84, 86).

Caballero, Jose, Christian Ledig, Andrew Aitken, Alejandro Acosta, Johannes Totz, Zehan Wang, and Wenzhe Shi (July 2017). "Real-Time Video Super-Resolution With Spatio-Temporal Networks and Motion Compensation". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2848–2857 (cit. on p. 64).

Carreira, João, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman (2018). "A Short Note about Kinetics-600". In: arXiv: 1808.01340 (cit. on p. 140).

Castrejon, Lluis, Nicolas Ballas, and Aaron Courville (Oct. 2019). "Improved Conditional VRNNs for Video Prediction". In: *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7607–7616 (cit. on pp. 65, 140).

Chang, Bo, Minmin Chen, Eldad Haber, and Ed H. Chi (2019). "AntisymmetricRNN: A Dynamical System View on Recurrent Neural Networks". In: *International Conference on Learning Representations* (cit. on p. 17).

Chang, Ya-Liang, Zhe Yu Liu, Kuan-Ying Lee, and Winston Hsu (Oct. 2019). "Free-Form Video Inpainting With 3D Gated Convolution and Temporal PatchGAN". In: *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9065–9074 (cit. on p. 19).

Chang, Shiyu, Yang Zhang, Wei Han, Mo Yu, Xiaoxiao Guo, Wei Tan, Xiaodong Cui, Michael Witbrock, Mark A. Hasegawa-Johnson, and Thomas S. Huang (2017). "Dilated Recurrent Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna Wallach, Rob Fergus, Swaminathan Venkata Narayana Vishwanathan, and Roman Garnett. Vol. 30. Curran Associates, Inc., pp. 76–86 (cit. on p. 14).

Chen, Lili, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch (2021). "Decision Transformer: Reinforcement Learning via Sequence Modeling". In: arXiv: 2106.01345 (cit. on p. 20).

Chen, Lin and Sheng Xu (2021). "Deep Neural Tangent Kernel and Laplace Kernel Have the Same RKHS". In: *International Conference on Learning Representations* (cit. on pp. 104, 240).

Chen, Mickaël (July 2020). "Learning with Weak Supervision Using Deep Generative Networks". PhD thesis. Sorbonne Université. URL: https://tel.archives-ouvertes.fr/tel-03153266 (cit. on p. 5).

Chen, Ricky T. Q., Brandon Amos, and Maximilian Nickel (2021). "Learning Neural Event Functions for Ordinary Differential Equations". In: *International Conference on Learning Representations* (cit. on p. 138).

Chen, Ricky T. Q., Yulia Rubanova, Jesse Bettencourt, and David Duvenaud (2018). "Neural Ordinary Differential Equations". In: *Advances in Neural Information Processing Systems*. Ed. by Samy Bengio, Hanna Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett. Vol. 31. Curran Associates, Inc., pp. 6572–6583 (cit. on pp. 7, 16, 17, 22, 66, 74, 89).

Chen, Ting, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton (July 2020). "A Simple Framework for Contrastive Learning of Visual Representations". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 1597–1607 (cit. on p. 21).

Chen, Xi, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel (2016). "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. Ed. by Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett. Vol. 29. Curran Associates, Inc., pp. 2172–2180 (cit. on p. 197).

Chen, Zhengdao, Jianyu Zhang, Martin Arjovsky, and Léon Bottou (2020). "Symplectic Recurrent Neural Networks". In: *International Conference on Learning Representations* (cit. on pp. 84, 198).

Cheng, Xiuyuan and Yao Xie (2021). "Neural Tangent Kernel Maximum Mean Discrepancy". In: arXiv: 2106.03227 (cit. on p. 113).

Child, Rewon, Scott Gray, Alec Radford, and Ilya Sutskever (2019). "Generating Long Sequences with Sparse Transformers". In: arXiv: 1904.10509 (cit. on p. 57).

Cho, Kyunghyun, Bart van Merriënboer, 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: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, pp. 1724–1734 (cit. on p. 14).

Chu, Casey, Kentaro Minami, and Kenji Fukumizu (2020). "The equivalence between Stein variational gradient descent and black-box variational inference". In: arXiv: 2004.01822 (cit. on p. 105).

Chung, Junyoung, Sungjin Ahn, and Yoshua Bengio (2017). "Hierarchical Multiscale Recurrent Neural Networks". In: *International Conference on Learning Representations* (cit. on p. 14).

Chung, Junyoung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron Courville, and Yoshua Bengio (2015). "A Recurrent Latent Variable Model for Sequential Data". In: *Advances in Neural Information Processing Systems*. Ed. by Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett. Vol. 28. Curran Associates, Inc., pp. 2980–2988 (cit. on pp. 31, 33, 65).

Collobert, Ronan and Jason Weston (2008). "A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning". In: *Proceedings of the 25th International Conference on Machine Learning*. ICML'08. Helsinki, Finland: Association for Computing Machinery, pp. 160–167 (cit. on p. 19).

Corless, Robert M., Hui Ding, Nicholas J. Higham, and David J. Jeffrey (2007). "The Solution of $S\exp(S) = A$ is Not Always the Lambert W Function of A". In: *Proceedings of the 2007 International Symposium on Symbolic and Algebraic Computation*. ISSAC'07. Waterloo, Ontario, Canada: Association for Computing Machinery, pp. 116–121 (cit. on p. 235).

Corless, Robert M., Gaston H. Gonnet, David E. G. Hare, David J. Jeffrey, and Donald E. Knuth (Dec. 1996). "On the Lambert W function". In: *Advances in Computational Mathematics* 5.1, pp. 329–359 (cit. on p. 234).

Cui, Zhicheng, Wenlin Chen, and Yixin Chen (2016). "Multi-Scale Convolutional Neural Networks for Time Series Classification". In: arXiv: 1603.06995 (cit. on p. 19).

Cuturi, Marco and Mathieu Blondel (Aug. 2017). "Soft-DTW: a Differentiable Loss Function for Time-Series". 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. PMLR, pp. 894–903 (cit. on p. 30).

Dai, Bin and David Wipf (2019). "Diagnosing and Enhancing VAE Models". In: *International Conference on Learning Representations* (cit. on p. 26).

Dau, Hoang Anh, Eamonn Keogh, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, Yanping, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista (Oct. 2018). *The UCR Time Series Classification Archive*. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/ (cit. on pp. 39, 47, 145).

De Brabandere, Bert, Xu Jia, Tinne Tuytelaars, and Luc Van Gool (2016). "Dynamic Filter Networks". In: *Advances in Neural Information Processing Systems*. Ed. by

Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett. Vol. 29. Curran Associates, Inc., pp. 667–675 (cit. on p. 64).

De Brouwer, Edward, Jaak Simm, Adam Arany, and Yves Moreau (2019). "GRU-ODE-Bayes: Continuous Modeling of Sporadically-Observed Time Series". In: *Advances in Neural Information Processing Systems*. Ed. by Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily Fox, and Roman Garnett. Vol. 32. Curran Associates, Inc., pp. 7379–7390 (cit. on pp. 17, 66).

Avila Belbute-Peres, Filipe de, Kevin A. Smith, Kelsey R. Allen, Joshua B. Tenenbaum, and J. Zico Kolter (2018). "End-to-End Differentiable Physics for Learning and Control". In: *Advances in Neural Information Processing Systems*. Ed. by Samy Bengio, Hanna Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett. Vol. 31. Curran Associates, Inc., pp. 7178–7189 (cit. on p. 84).

Bézenac, Emmanuel de, Ibrahim Ayed, and Patrick Gallinari (2021). "CycleGAN Through the Lens of (Dynamical) Optimal Transport". In: *Machine Learning and Knowledge Discovery in Databases. Research Track*. Ed. by Nuria Oliver, Fernando Pérez-Cruz, Stefan Kramer, Jesse Read, and Jose A. Lozano. Cham, Switzerland: Springer International Publishing, pp. 132–147 (cit. on p. 70).

Bézenac, Emmanuel de, Arthur Pajot, and Patrick Gallinari (2018). "Deep Learning for Physical Processes: Incorporating Prior Scientific Knowledge". In: *International Conference on Learning Representations* (cit. on pp. 17, 84, 93, 199, 201, 202).

Bézenac, Emmanuel de, Syama Sundar Rangapuram, Konstantinos Benidis, Michael Bohlke-Schneider, Richard Kurle, Lorenzo Stella, Hilaf Hasson, Patrick Gallinari, and Tim Januschowski (2020). "Normalizing Kalman Filters for Multivariate Time Series Analysis". In: *Advances in Neural Information Processing Systems*. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. Vol. 33. Curran Associates, Inc., pp. 2995–3007 (cit. on p. 33).

Delasalles, Edouard, Sylvain Lamprier, and Ludovic Denoyer (Nov. 2019). "Learning Dynamic Author Representations with Temporal Language Models". In: *IEEE International Conference on Data Mining (ICDM)*, pp. 120–129 (cit. on p. 6).

Demailly, Jean-Pierre (2006). *Analyse numérique et équations différentielles*. 3rd edition. Grenoble Sciences. EDP Sciences (cit. on p. 15).

Demšar, Janez, Tomaž Curk, Aleš Erjavec, Črt Gorup, Tomaž Hočevar, Mitar Milutinovič, Martin Možina, Matija Polajnar, Marko Toplak, Anže Starič, Miha Štajdohar, Lan Umek, Lan Žagar, Jure Žbontar, Marinka Žitnik, and Blaž Zupan (2013). "Orange: Data Mining Toolbox in Python". In: *Journal of Machine Learning Research* 14.35, pp. 2349–2353 (cit. on p. 48).

Denton, Emily and Vighnesh Birodkar (2017). "Unsupervised Learning of Disentangled Representations from Video". In: *Advances in Neural Information Processing Systems*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna Wallach, Rob Fergus, Swaminathan Venkata Narayana Vishwanathan, and Roman Garnett. Vol. 30. Curran Associates, Inc., pp. 4417–4426 (cit. on pp. 23, 39, 40, 69, 83, 95, 96, 196, 200, 201).

Denton, Emily and Rob Fergus (July 2018). "Stochastic Video Generation with a Learned Prior". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine

Learning Research. Stockholmsmässan, Stockholm, Sweden: PMLR, pp. 1174–1183 (cit. on pp. 63–65, 71, 72, 91, 93, 140, 167–169, 201, 205).

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (June 2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota, United States of America: Association for Computational Linguistics, pp. 4171–4186 (cit. on pp. 6, 19, 28).

Dhariwal, Prafulla, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever (2020). "Jukebox: A Generative Model for Music". In: arXiv: 2005.00341 (cit. on p. 30).

Dheeru, Dua and Efi Karra Taniskidou (2017). *UCI Machine Learning Repository*. URL: http://archive.ics.uci.edu/ml (cit. on p. 53).

Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio (2017). "Density estimation using Real NVP". In: *International Conference on Learning Representations* (cit. on p. 29).

Donà, Jérémie, Jean-Yves Franceschi, Sylvain Lamprier, and Patrick Gallinari (2021). "PDE-Driven Spatiotemporal Disentanglement". In: *International Conference on Learning Representations* (cit. on pp. 9, 61).

Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby (2021). "An Image is Worth 16×16 Words: Transformers for Image Recognition at Scale". In: *International Conference on Learning Representations* (cit. on p. 20).

Dosovitskiy, Alexey, Philipp Fischer, Eddy Ilg, Philip Häusser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox (Dec. 2015). "FlowNet: Learning Optical Flow With Convolutional Networks". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 2758–2766 (cit. on pp. 17, 83).

Dosovitskiy, Alexey, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox (2014). "Discriminative Unsupervised Feature Learning with Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger. Vol. 27. Curran Associates, Inc., pp. 766–774 (cit. on p. 46).

Dugas, Charles, Yoshua Bengio, François Bélisle, Claude Nadeau, and René Garcia (2001). "Incorporating Second-Order Functional Knowledge for Better Option Pricing". In: *Advances in Neural Information Processing Systems*. Ed. by Todd K. Leen, Thomas G. Dietterich, and Volker Tresp. Vol. 13. MIT Press, pp. 451–457 (cit. on p. 170).

Duncan, Andrew, Nikolas Nüsken, and Lukasz Szpruch (2019). "On the geometry of Stein variational gradient descent". In: arXiv: 1912.00894 (cit. on p. 111).

Dupont, Emilien, Arnaud Doucet, and Yee Whye Teh (2019). "Augmented Neural ODEs". In: *Advances in Neural Information Processing Systems*. Ed. by Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily Fox, and Roman Garnett. Vol. 32. Curran Associates, Inc., pp. 3140–3150 (cit. on p. 17).

Ebert, Frederik, Chelsea Finn, Alex X. Lee, and Sergey Levine (Nov. 2017). "Self-Supervised Visual Planning with Temporal Skip Connections". In: *Proceedings of the 1st Annual Conference on Robot Learning*. Ed. by Sergey Levine, Vincent Vanhoucke, and Ken Goldberg. Vol. 78. Proceedings of Machine Learning Research. PMLR, pp. 344–356 (cit. on pp. 78, 168).

Fan, Hehe, Linchao Zhu, and Yi Yang (July 2019). "Cubic LSTMs for Video Prediction". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 1, pp. 8263–8270 (cit. on p. 64).

Fan, Zhou and Zhichao Wang (2020). "Spectra of the Conjugate Kernel and Neural Tangent Kernel for linear-width neural networks". In: *Advances in Neural Information Processing Systems*. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. Vol. 33. Curran Associates, Inc., pp. 7710–7721 (cit. on pp. 104, 239).

Farkas, Bálint and Sven-Ake Wegner (2016). "Variations on Barbălat's Lemma". In: *The American Mathematical Monthly* 123.8, pp. 825–830 (cit. on p. 229).

Feydy, Jean, Thibault Séjourné, François-Xavier Vialard, Shun-ichi Amari, Alain Trouve, and Gabriel Peyré (Apr. 2019). "Interpolating between Optimal Transport and MMD using Sinkhorn Divergences". In: *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*. Ed. by Kamalika Chaudhuri and Masashi Sugiyama. Vol. 89. Proceedings of Machine Learning Research. PMLR, pp. 2681–2690 (cit. on pp. 116, 120, 241).

Finn, Chelsea, Ian Goodfellow, and Sergey Levine (2016). "Unsupervised Learning for Physical Interaction through Video Prediction". In: *Advances in Neural Information Processing Systems*. Ed. by Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett. Vol. 29. Curran Associates, Inc., pp. 64–72 (cit. on pp. 6, 17, 64, 65).

Finn, Chelsea, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, and Pieter Abbeel (May 2016). "Deep spatial autoencoders for visuomotor learning". In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 512–519 (cit. on p. 83).

Fortuin, Vincent, Matthias Hüser, Francesco Locatello, Heiko Strathmann, and Gunnar Rätsch (2019). "SOM-VAE: Interpretable Discrete Representation Learning on Time Series". In: *International Conference on Learning Representations* (cit. on p. 40).

Fourier, Jean Baptiste Joseph (1822). *Théorie analytique de la chaleur*. Didot, Firmin (cit. on p. 84).

Fraccaro, Marco (2018). "Deep Latent Variable Models for Sequential Data". PhD thesis. Danmarks Tekniske Universitet. URL: https://orbit.dtu.dk/en/publications/deep-latent-variable-models-for-sequential-data (cit. on p. 31).

Fraccaro, Marco, Simon Kamronn, Ulrich Paquet, and Ole Winther (2017). "A Disentangled Recognition and Nonlinear Dynamics Model for Unsupervised Learning". In: *Advances in Neural Information Processing Systems 30*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna Wallach, Rob Fergus, Swaminathan Venkata Narayana Vishwanathan, and Roman Garnett. Vol. 30. Curran Associates, Inc., pp. 3604–3613 (cit. on pp. 22, 33, 65, 172, 173).

Fraccaro, Marco, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther (2016). "Sequential Neural Models with Stochastic Layers". In: *Advances in Neural Information Processing Systems 29*. Ed. by Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett. Vol. 29. Curran Associates, Inc., pp. 2207–2215 (cit. on p. 65).

Franceschi, Jean-Yves, Emmanuel de Bézenac, Ibrahim Ayed, Mickaël Chen, Sylvain Lamprier, and Patrick Gallinari (2021). "A Neural Tangent Kernel Perspective of GANs". In: arXiv: 2106.05566 (cit. on pp. 10, 101).

Franceschi, Jean-Yves, Edouard Delasalles, Mickaël Chen, Sylvain Lamprier, and Patrick Gallinari (July 2020). "Stochastic Latent Residual Video Prediction". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 3233–3246 (cit. on pp. 9, 61).

Franceschi, Jean-Yves, Aymeric Dieuleveut, and Martin Jaggi (2019). "Unsupervised Scalable Representation Learning for Multivariate Time Series". In: *Advances in Neural Information Processing Systems*. Ed. by Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily Fox, and Roman Garnett. Vol. 32. Curran Associates, Inc., pp. 4650–4661 (cit. on pp. 8, 37).

Fubini, Guido (1907). "Sugli integrali multipli". In: *Accademia dei Lincei, Rendiconti, V. Serie* 16.1, pp. 608–614 (cit. on p. 218).

Funahashi, Ken-ichi and Yuichi Nakamura (1993). "Approximation of Dynamical Systems by Continuous Time Recurrent Neural Networks". In: *Neural Networks* 6.6, pp. 801–806 (cit. on p. 16).

Ganea, Octavian-Eugen, Gary Bécigneul, and Thomas Hofmann (2018). "Hyperbolic Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by Samy Bengio, Hanna Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett. Vol. 31. Curran Associates, Inc., pp. 5350–5360 (cit. on p. 14).

Gao, Hang, Huazhe Xu, Qi-Zhi Cai, Ruth Wang, Fisher Yu, and Trevor Darrell (Oct. 2019). "Disentangling Propagation and Generation for Video Prediction". In: *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9005–9014 (cit. on p. 64).

Gehring, Jonas, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin (Aug. 2017). "Convolutional Sequence to Sequence Learning". 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. PMLR, pp. 1243–1252 (cit. on p. 19).

Geiger, Mario, Stefano Spigler, Arthur Jacot, and Matthieu Wyart (Nov. 2020). "Disentangling feature and lazy training in deep neural networks". In: *Journal of Statistical Mechanics: Theory and Experiment* 2020.11 (cit. on p. 236).

Geneva, Nicholas and Nicholas Zabaras (2020). "Transformers for Modeling Physical Systems". In: arXiv: 2010.03957 (cit. on p. 20).

Gers, Felix A., Jürgen Schmidhuber, and Fred Cummins (2000). "Learning to Forget: Continual Prediction with LSTM". In: *Neural Computation* 12.10, pp. 2451–2471 (cit. on p. 12).

Gholaminejad, Amir, Kurt Keutzer, and George Biros (July 2019). "ANODE: Unconditionally Accurate Memory-Efficient Gradients for Neural ODEs". In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence — IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, pp. 730–736 (cit. on p. 17).

Goldberg, Yoav and Omer Levy (2014). "word2vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method". In: arXiv: 1402.3722 (cit. on p. 41).

Gomez, Aidan N., Mengye Ren, Raquel Urtasun, and Roger B. Grosse (2017). "The Reversible Residual Network: Backpropagation Without Storing Activations". In: *Advances in Neural Information Processing Systems*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna Wallach, Rob Fergus, Swaminathan Venkata Narayana Vishwanathan, and Roman Garnett. Vol. 30. Curran Associates, Inc., pp. 2211–2221 (cit. on p. 17).

Goodfellow, Ian (2016). "NIPS 2016 Tutorial: Generative Adversarial Networks". In: arXiv: 1701.00160 (cit. on pp. 24, 27, 65, 107).

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. Adaptive Computation and Machine Learning. http://www.deeplearningbook.org. MIT Press (cit. on pp. 5, 11, 12, 19).

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2014). "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. Ed. by Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger. Vol. 27. Curran Associates, Inc., pp. 2672–2680 (cit. on pp. 26, 27, 104–106).

Grathwohl, Will, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud (2019). "FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models". In: *International Conference on Learning Representations* (cit. on pp. 17, 29).

Graves, Alex (2013). "Generating Sequences With Recurrent Neural Networks". In: arXiv: 1308.0850 (cit. on pp. 31, 64).

Graves, Alex and Jürgen Schmidhuber (2005). "Framewise phoneme classification with bidirectional LSTM and other neural network architectures". In: *Neural Networks* 18.5. IJCNN 2005, pp. 602–610 (cit. on pp. 12, 14).

Greff, Klaus, Rupesh K. Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber (2017). "LSTM: A Search Space Odyssey". In: *IEEE Transactions on Neural Networks and Learning Systems* 28.10, pp. 2222–2232 (cit. on pp. 11, 14).

Gregor, Karol, George Papamakarios, Frederic Besse, Lars Buesing, and Théophane Weber (2019). "Temporal Difference Variational Auto-Encoder". In: *International Conference on Learning Representations* (cit. on pp. 22, 33, 63).

Gretton, Arthur, Karsten M. Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex Smola (2007). "A Kernel Method for the Two-Sample-Problem". In: *Advances in Neural Information Processing Systems*. Ed. by Bernhard Schölkopf, John C. Platt, and Thomas Hoffman. Vol. 19. MIT Press, pp. 513–520 (cit. on p. 112).

Gretton, Arthur, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola (2012). "A Kernel Two-Sample Test". In: *Journal of Machine Learning Research* 13.25, pp. 723–773 (cit. on pp. 112, 139).

Greydanus, Sam, Stefan Lee, and Alan Fern (2021). "Piecewise-constant Neural ODEs". In: arXiv: 2106.06621 (cit. on p. 138).

Greydanus, Samuel, Misko Dzamba, and Jason Yosinski (2019). "Hamiltonian Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily Fox, and Roman Garnett. Vol. 32. Curran Associates, Inc., pp. 15379–15389 (cit. on pp. 22, 84).

Grönwall, Thomas Hakon (1919). "Note on the Derivatives with Respect to a Parameter of the Solutions of a System of Differential Equations". In: *Annals of Mathematics* 20.4, pp. 292–296 (cit. on p. 215).

Gulrajani, Ishaan, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville (2017). "Improved Training of Wasserstein GANs". In: *Advances in Neural Information Processing Systems*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna Wallach, Rob Fergus, Swaminathan Venkata Narayana Vishwanathan, and Roman Garnett. Vol. 30. Curran Associates, Inc., pp. 5769–5779 (cit. on p. 27).

Ha, Jung-Su, Young-Jin Park, Hyeok-Joo Chae, Soon-Seo Park, and Han-Lim Choi (2018). "Adaptive Path-Integral Autoencoders: Representation Learning and Planning for Dynamical Systems". In: *Advances in Neural Information Processing Systems*. Ed. by Samy Bengio, Hanna Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett. Vol. 31. Curran Associates, Inc., pp. 8941–8952 (cit. on p. 18).

Haber, Eldad and Lars Ruthotto (Dec. 2018). "Stable architectures for deep neural networks". In: *Inverse Problems* 34.1 (cit. on p. 17).

Hafner, Danijar, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson (June 2019). "Learning Latent Dynamics for Planning from Pixels". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, United States of America: PMLR, pp. 2555–2565 (cit. on p. 65).

Hairer, Ernst, Syvert P. Nørsett, and Gerhard Wanner (1993). "Solving Ordinary Differential Equations I. Nonstiff Problems". In: Springer Series in Computational Mathematics. Berlin - Heidelberg, Germany: Springer Berlin Heidelberg. Chap. Runge-Kutta and Extrapolation Methods, pp. 129–353 (cit. on p. 199).

Hamilton, William Rowan (1835). "Second Essay on a General Method in Dynamics". In: *Philosophical Transactions of the Royal Society* 125, pp. 95–144 (cit. on pp. 83, 84).

Han, Tengda, Weidi Xie, and Andrew Zisserman (2020). "Self-supervised Co-Training for Video Representation Learning". In: *Advances in Neural Information Processing Systems*. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. Vol. 33. Curran Associates, Inc., pp. 5679–5690 (cit. on p. 21).

Hasani, Ramin, Mathias Lechner, Alexander Amini, Daniela Rus, and Radu Grosu (May 2021). "Liquid Time-constant Networks". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.9, pp. 7657–7666 (cit. on p. 16).

He, Jiawei, Andreas Lehrmann, Joseph Marino, Greg Mori, and Leonid Sigal (Sept. 2018). "Probabilistic Video Generation using Holistic Attribute Control". In: *The European Conference on Computer Vision (ECCV)*. Ed. by Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss. Springer International Publishing, pp. 466–483 (cit. on p. 65).

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (June 2016). "Deep Residual Learning for Image Recognition". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778 (cit. on pp. 17, 235).

Higgins, Irina, David Amos, David Pfau, Sébastien Racanière, Loïc Matthey, Danilo Jimenez Rezende, and Alexander Lerchner (2018). "Towards a Definition of Disentangled Representations". In: arXiv: 1812.02230 (cit. on p. 22).

Higgins, Irina, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner (2017). "β-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". In: *International Conference on Learning Representations* (cit. on pp. 26, 170).

Higham, Nicholas J. (2008). *Functions of Matrices: Theory and Computation*. Society for Industrial and Applied Mathematics (cit. on p. 230).

Hochreiter, Sepp and Jürgen Schmidhuber (Nov. 1997). "Long Short-Term Memory". In: *Neural computation* 9.8, pp. 1735–1780 (cit. on pp. 6, 12).

Horn, Berthold K. P. and Brian G. Schunck (Aug. 1981). "Determining Optical Flow". In: *Artificial Intelligence* 17.1–3, pp. 185–203 (cit. on pp. 17, 83).

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White (1989). "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5, pp. 359–366 (cit. on p. 239).

Hron, Jiri, Yasaman Bahri, Jascha Sohl-Dickstein, and Roman Novak (July 2020). "Infinite attention: NNGP and NTK for deep attention networks". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 4376–4386 (cit. on p. 236).

Hsieh, Jun-Ting, Bingbin Liu, De-An Huang, Li Fei-Fei, and Juan Carlos Niebles (2018). "Learning to Decompose and Disentangle Representations for Video Prediction". In: *Advances in Neural Information Processing Systems*. Ed. by Samy Bengio, Hanna Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett. Vol. 31. Curran Associates, Inc., pp. 515–524 (cit. on pp. 22, 23, 95, 201).

Hsu, Wei-Ning, Yu Zhang, and James Glass (2017). "Unsupervised Learning of Disentangled and Interpretable Representations from Sequential Data". In: *Advances in Neural Information Processing Systems*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna Wallach, Rob Fergus, Swaminathan Venkata Narayana Vishwanathan, and Roman Garnett. Vol. 30. Curran Associates, Inc., pp. 1876–1887 (cit. on p. 23).

Huang, Kaixuan, Yuqing Wang, Molei Tao, and Tuo Zhao (2020). "Why Do Deep Residual Networks Generalize Better than Deep Feedforward Networks? — A Neural Tangent Kernel Perspective". In: *Advances in Neural Information Processing Systems*. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. Vol. 33. Curran Associates, Inc., pp. 2698–2709 (cit. on p. 225).

Hyvarinen, Aapo and Hiroshi Morioka (2016). "Unsupervised Feature Extraction by Time-Contrastive Learning and Nonlinear ICA". In: *Advances in Neural Information Processing Systems*. Ed. by Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett. Vol. 29. Curran Associates, Inc., pp. 3772–3780 (cit. on pp. 21, 40).

Iacono, Roberto and John P. Boyd (2017). "New approximations to the principal real-valued branch of the Lambert W-function". In: *Advances in Computational Mathematics* 43.6, pp. 1403–1436 (cit. on p. 235).

Ionescu, Catalin, Fuxin Li, and Cristian Sminchisescu (Nov. 2011). "Latent Structured Models for Human Pose Estimation". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 2220–2227 (cit. on pp. 77, 168).

Ionescu, Catalin, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu (July 2014). "Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.7, pp. 1325–1339 (cit. on pp. 77, 168).

Ismail Fawaz, Hassan, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller (2019). "Deep learning for time series classification: a review". In: *Data Mining and Knowledge Discovery* 33, pp. 917–963 (cit. on pp. 19, 44, 48).

Jacot, Arthur, Franck Gabriel, François Ged, and Clément Hongler (2019). "Order and Chaos: NTK views on DNN Normalization, Checkerboard and Boundary Artifacts". In: arXiv: 1907.05715 (cit. on p. 105).

Jacot, Arthur, Franck Gabriel, and Clement Hongler (2018). "Neural Tangent Kernel: Convergence and Generalization in Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by Samy Bengio, Hanna Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett. Vol. 31. Curran Associates, Inc., pp. 8580–8589 (cit. on pp. 18, 103, 104, 108, 113, 114, 225, 239, 241).

Janner, Michael, Qiyang Li, and Sergey Levine (2021). "Reinforcement Learning as One Big Sequence Modeling Problem". In: arXiv: 2106.02039 (cit. on p. 20).

Jansen, Aren, Manoj Plakal, Ratheet Pandya, Daniel P. W. Ellis, Shawn Hershey, Jiayang Liu, R. Channing Moore, and Rif A. Saurous (Apr. 2018). "Unsupervised Learning of Semantic Audio Representations". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 126–130 (cit. on pp. 21, 40).

Jaques, Miguel, Michael Burke, and Timothy Hospedales (2020). "Physics-as-Inverse-Graphics: Unsupervised Physical Parameter Estimation from Video". In: *International Conference on Learning Representations* (cit. on p. 23).

Jebara, Tony (2008). *Machine Learning. Discriminative and Generative*. The International Series in Engineering and Computer Science. Boston, Massachusetts, United States of America: Springer US (cit. on p. 23).

Ji, Ziwei, Matus Telgarsky, and Ruicheng Xian (2020). "Neural tangent kernels, transportation mappings, and universal approximation". In: *International Conference on Learning Representations* (cit. on pp. 121, 240).

Jia, Huabing, Wei Xu, Xiaoshan Zhao, and Zhanguo Li (Mar. 2008). "Separation of variables and exact solutions to nonlinear diffusion equations with x-dependent

convection and absorption". In: *Journal of Mathematical Analysis and Applications* 339.2, pp. 982–995 (cit. on p. 85).

Jia, Junteng and Austin R. Benson (2019). "Neural Jump Stochastic Differential Equations". In: *Advances in Neural Information Processing Systems*. Ed. by Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily Fox, and Roman Garnett. Vol. 32. Curran Associates, Inc., pp. 9847–9858 (cit. on p. 31).

Jiang, Huaizu, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz (June 2018). "Super SloMo: High Quality Estimation of Multiple Intermediate Frames for Video Interpolation". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9000–9008 (cit. on p. 64).

Jin, Beibei, Yu Hu, Qiankun Tang, Jingyu Niu, Zhiping Shi, Yinhe Han, and Xiaowei Li (June 2020). "Exploring Spatial-Temporal Multi-Frequency Analysis for High-Fidelity and Temporal-Consistency Video Prediction". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4553–4562 (cit. on p. 64).

Jouppi, Norman P., Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Daniel Killebrew, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon (2017). "In-Datacenter Performance Analysis of a Tensor Processing Unit". In: *Proceedings of the 44th Annual International Symposium on Computer Architecture*. ISCA'17. Toronto, Ontario, Canada: Association for Computing Machinery, pp. 1–12 (cit. on p. 5).

Kalchbrenner, Nal, Aäron van den Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu (Aug. 2017). "Video Pixel 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. 1771–1779 (cit. on pp. 6, 65).

Kalman, Rudolf E. (Mar. 1960). "A New Approach to Linear Filtering and Prediction Problems". In: *Journal of Basic Engineering* 82.1, pp. 35–45 (cit. on p. 33).

Kalnins, Ernie G., Willard Miller Jr., and Graeme C. Williams (1992). "Recent Advances in the Use of Separation of Variables Methods in General Relativity". In: *Philosophical Transactions: Physical Sciences and Engineering* 340.1658, pp. 337–352 (cit. on p. 85).

Karl, Maximilian, Maximilian Soelch, Justin Bayer, and Patrick van der Smagt (2017). "Deep Variational Bayes Filters: Unsupervised Learning of State Space Models from

Raw Data". In: *International Conference on Learning Representations* (cit. on pp. 22, 33, 63, 65, 68, 172, 173).

Karras, Tero, Samuli Laine, and Timo Aila (June 2019). "A Style-Based Generator Architecture for Generative Adversarial Networks". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4396–4405 (cit. on p. 113).

Karras, Tero, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila (June 2020). "Analyzing and Improving the Image Quality of StyleGAN". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8107–8116 (cit. on pp. 6, 28).

Katharopoulos, Angelos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret (July 2020). "Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 5156–5165 (cit. on p. 20).

Kingma, Diederik P. and Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations* (cit. on pp. 46, 146, 170, 204).

Kingma, Diederik P. and Prafulla Dhariwal (2018). "Glow: Generative Flow with Invertible 1×1 Convolutions". In: *Advances in Neural Information Processing Systems*. Ed. by Samy Bengio, Hanna Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett. Vol. 31. Curran Associates, Inc., pp. 10236–10245 (cit. on pp. 29, 65).

Kingma, Diederik P. and Max Welling (2014). "Auto-Encoding Variational Bayes". In: *International Conference on Learning Representations* (cit. on pp. 25, 70, 165).

– (2019). "An Introduction to Variational Autoencoders". In: *Foundations and Trends® in Machine Learning* 12.4, pp. 307–392 (cit. on pp. 6, 26).

Kloeden, Peter E. and Eckhard Platen (1992). "Introduction to Stochastic Time Discrete Approximation". In: *Numerical Solution of Stochastic Differential Equations*. Applications of Mathematics. Berlin - Heidelberg, Germany: Springer Berlin Heidelberg, pp. 305–337 (cit. on p. 68).

Kobyzev, Ivan, Simon J. D. Prince, and Marcus A. Brubaker (2020). "Normalizing Flows: An Introduction and Review of Current Methods". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.11, pp. 3964–3979 (cit. on p. 28).

Kosiorek, Adam R., Hyunjik Kim, Yee Whye Teh, and Ingmar Posner (2018). "Sequential Attend, Infer, Repeat: Generative Modelling of Moving Objects". In: *Advances in Neural Information Processing Systems*. Ed. by Samy Bengio, Hanna Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett. Vol. 31. Curran Associates, Inc., pp. 8615–8625 (cit. on p. 23).

Kovachki, Nikola B. and Andrew M. Stuart (2021). "Continuous Time Analysis of Momentum Methods". In: *Journal of Machine Learning Research* 22.17, pp. 1–40 (cit. on p. 7).

Kraskov, Alexander, Harald Stögbauer, and Peter Grassberger (June 2004). "Estimating mutual information". In: *Physical Review E* 69 (6) (cit. on p. 197).

Krishnan, Rahul G., Uri Shalit, and David Sontag (2015). "Deep Kalman filters". In: arXiv: 1511.05121 (cit. on p. 33).

Krishnan, Rahul G., Uri Shalit, and David Sontag (2017). "Structured Inference Networks for Nonlinear State Space Models". In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. AAAI'17. San Francisco, California, United States of America: AAAI Press, pp. 2101–2109 (cit. on pp. 31, 32, 63, 65).

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by Fernando Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger. Vol. 25. Curran Associates, Inc., pp. 1097–1105 (cit. on pp. 5, 19).

Kullback, Solomon and Richard A. Leibler (1951). "On Information and Sufficiency". In: *The Annals of Mathematical Statistics* 22.1, pp. 79–86 (cit. on p. 24).

Kumar, Manoj, Mohammad Babaeizadeh, Dumitru Erhan, Chelsea Finn, Sergey Levine, Laurent Dinh, and Diederik P. Kingma (2020). "VideoFlow: A Conditional Flow-Based Model for Stochastic Video Generation". In: *International Conference on Learning Representations* (cit. on pp. 6, 65).

Kurach, Karol, Mario Lucic, Xiaohua Zhai, Marcin Michalski, and Sylvain Gelly (June 2019). "A Large-Scale Study on Regularization and Normalization in GANs". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 3581–3590 (cit. on p. 103).

Kutta, Martin Wilhelm (1901). "Beitrag zur näherungweisen Integration totaler Differentialgleichungen". In: *Zeitschrift für Mathematik und Physik* 45, pp. 435–453 (cit. on p. 199).

Lample, Guillaume, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato (2018). "Unsupervised Machine Translation Using Monolingual Corpora Only". In: *International Conference on Learning Representations* (cit. on p. 6).

Larochelle, Hugo and Iain Murray (Apr. 2011). "The Neural Autoregressive Distribution Estimator". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Geoffrey Gordon, David Dunson, and Miroslav Dudík. Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, Florida, United States of America: PMLR, pp. 29–37 (cit. on p. 28).

Le Dret, Hervé and Brigitte Lucquin (2016). "Partial Differential Equations: Modeling, Analysis and Numerical Approximation". In: International Series of Numerical Mathematics. Cham, Switzerland: Springer International Publishing. Chap. The Heat Equation, pp. 219–251 (cit. on pp. 16, 85, 194).

Le Guen, Vincent and Nicolas Thome (2019). "Shape and Time Distortion Loss for Training Deep Time Series Forecasting Models". In: *Advances in Neural Information Processing Systems*. Ed. by Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily Fox, and Roman Garnett. Vol. 32. Curran Associates, Inc., pp. 4189–4201 (cit. on p. 30).

– (June 2020). "Disentangling Physical Dynamics from Unknown Factors for Unsupervised Video Prediction". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11471–11481 (cit. on pp. 23, 66, 91, 93, 98, 201).

Cun, Yann le (1988). "A Theoretical Framework for Back-Propagation". In: *Proceedings of the 1988 Connectionist Models Summer School*. Ed. by David S. Touretzky,

Geoffrey E. Hinton, and Terry Sejnowski. Carnegie Mellon University, Pittsburgh, Pennsylvania, United States of America: Morgan Kaufmann, pp. 21–28 (cit. on p. 17).

LeCun, Yann, Bernhard Boser, John S. Denker, Donnie Henderson, Rich E. Howard, Wayne Hubbard, and Lawrence D. Jackel (1989). "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Computation* 1.4, pp. 541–551 (cit. on p. 19).

LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner (Nov. 1998). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11, pp. 2278–2324 (cit. on pp. 72, 121, 167).

Lee, Alex X., Anusha Nagabandi, Pieter Abbeel, and Sergey Levine (2020). "Stochastic Latent Actor-Critic: Deep Reinforcement Learning with a Latent Variable Model". In: *Advances in Neural Information Processing Systems*. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. Vol. 33. Curran Associates, Inc., pp. 741–752 (cit. on p. 22).

Lee, Alex X., Richard Zhang, Frederik Ebert, Pieter Abbeel, Chelsea Finn, and Sergey Levine (2018). "Stochastic Adversarial Video Prediction". In: arXiv: 1804.01523 (cit. on pp. 19, 65, 71, 72, 169, 205).

Lee, Jaehoon, Yasaman Bahri, Roman Novak, Samuel S. Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein (2018). "Deep Neural Networks as Gaussian Processes". In: *International Conference on Learning Representations* (cit. on pp. 110, 223).

Lee, Jaehoon, Samuel S. Schoenholz, Jeffrey Pennington, Ben Adlam, Lechao Xiao, Roman Novak, and Jascha Sohl-Dickstein (2020). "Finite Versus Infinite Neural Networks: an Empirical Study". In: *Advances in Neural Information Processing Systems*. Ed. by Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily Fox, and Roman Garnett. Vol. 33. Curran Associates, Inc., pp. 15156–15172 (cit. on p. 104).

Lee, Jaehoon, Lechao Xiao, Samuel S. Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington (2019). "Wide Neural Networks of Any Depth Evolve as Linear Models Under Gradient Descent". In: *Advances in Neural Information Processing Systems*. Ed. by Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily Fox, and Roman Garnett. Vol. 32. Curran Associates, Inc., pp. 8572–8583 (cit. on p. 104).

Lei, Qi, Jinfeng Yi, Roman Vaculin, Lingfei Wu, and Inderjit S. Dhillon (July 2019). "Similarity Preserving Representation Learning for Time Series Clustering". In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence — IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, pp. 2845–2851 (cit. on p. 40).

Leipnik, Roy B. and Charles E. M. Pearce (2007). "The multivariate Faà di Bruno formula and multivariate Taylor expansions with explicit integral remainder term". In: *The ANZIAM Journal* 48.3, pp. 327–341 (cit. on p. 220).

Leshno, Moshe, Vladimir Ya. Lin, Allan Pinkus, and Shimon Schocken (1993). "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function". In: *Neural Networks* 6.6, pp. 861–867 (cit. on pp. 239, 240).

Li, Chun-Liang, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabas Poczos (2017). "MMD GAN: Towards Deeper Understanding of Moment Matching Network".

In: *Advances in Neural Information Processing Systems*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna Wallach, Rob Fergus, Swaminathan Venkata Narayana Vishwanathan, and Roman Garnett. Vol. 30. Curran Associates, Inc., pp. 2200–2210 (cit. on p. 113).

Li, Naihan, Shujie Liu, Yanqing Liu, Sheng Zhao, and Ming Liu (July 2019). "Neural Speech Synthesis with Transformer Network". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.1, pp. 6706–6713 (cit. on pp. 6, 20).

Li, Xuechen, Ting-Kam Leonard Wong, Ricky T. Q. Chen, and David Duvenaud (Aug. 2020). "Scalable Gradients for Stochastic Differential Equations". In: *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. Ed. by Silvia Chiappa and Roberto Calandra. Vol. 108. Proceedings of Machine Learning Research. PMLR, pp. 3870–3882 (cit. on p. 18).

Liang, Xiaodan, Lisa Lee, Wei Dai, and Eric P. Xing (Oct. 2017). "Dual Motion GAN for Future-Flow Embedded Video Prediction". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 1762–1770 (cit. on p. 64).

Liljefors, Felix, Moein Sorkhei, and Sofia Broomé (May 2020). "[Re] Unsupervised Scalable Representation Learning for Multivariate Time Series". Python. In: *ReScience C* 6.2 (cit. on p. 57).

Lim, Jae Hyun and Jong Chul Ye (2017). "Geometric GAN". In: arXiv: 1705.02894 (cit. on p. 105).

Lines, Jason and Anthony Bagnall (2015). "Time series classification with ensembles of elastic distance measures". In: *Data Mining and Knowledge Discovery* 29, pp. 565–592 (cit. on p. 47).

Lines, Jason, Sarah Taylor, and Anthony Bagnall (July 2018). "Time Series Classification with HIVE-COTE: The Hierarchical Vote Collective of Transformation-Based Ensembles". In: *ACM Transactions on Knowledge Discovery from Data* 12.5 (cit. on p. 47).

Lions, Jacques-Louis (1971). *Optimal Control of Systems Governed by Partial Differential Equations*. Vol. 170. Grundlehren der mathematischen Wissenschaften. Berlin - Heidelberg, Germany: Springer Verlag (cit. on p. 16).

Littwin, Etai, Tomer Galanti, Lior Wolf, and Greg Yang (2020). "On Infinite-Width Hypernetworks". In: *Advances in Neural Information Processing Systems*. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. Vol. 33. Curran Associates, Inc., pp. 13226–13237 (cit. on p. 105).

Littwin, Etai, Ben Myara, Sima Sabah, Joshua Susskind, Shuangfei Zhai, and Oren Golan (2020). "Collegial Ensembles". In: *Advances in Neural Information Processing Systems*. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. Vol. 33. Curran Associates, Inc., pp. 18738–18748 (cit. on p. 105).

Liu, Chaoyue, Libin Zhu, and Misha Belkin (2020). "On the linearity of large non-linear models: when and why the tangent kernel is constant". In: *Advances in Neural Information Processing Systems*. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. Vol. 33. Curran Associates, Inc., pp. 15954–15964 (cit. on pp. 104, 108, 235).

Liu, Ming-Yu, Xun Huang, Jiahui Yu, Ting-Chun Wang, and Arun Mallya (2021). "Generative Adversarial Networks for Image and Video Synthesis: Algorithms and Applications". In: *Proceedings of the IEEE* 109.5, pp. 839–862 (cit. on p. 27).

Liu, Qiang and Dilin Wang (2016). "Stein Variational Gradient Descent: A General Purpose Bayesian Inference Algorithm". In: *Advances in Neural Information Processing Systems*. Ed. by Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett. Vol. 29. Curran Associates, Inc., pp. 2378–2386 (cit. on pp. 111, 121).

Liu, Shuang, Olivier Bousquet, and Kamalika Chaudhuri (2017). "Approximation and Convergence Properties of Generative Adversarial Learning". In: *Advances in Neural Information Processing Systems*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna Wallach, Rob Fergus, Swaminathan Venkata Narayana Vishwanathan, and Roman Garnett. Vol. 30. Curran Associates, Inc., pp. 5551–5559 (cit. on p. 104).

Liu, Ziwei, Ping Luo, Xiaogang Wang, and Xiaoou Tang (Dec. 2015). "Deep Learning Face Attributes in the Wild". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 3730–3738 (cit. on p. 121).

Liu, Ziwei, Raymond A. Yeh, Xiaoou Tang, Yiming Liu, and Aseem Agarwala (Oct. 2017). "Video Frame Synthesis Using Deep Voxel Flow". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 4473–4481 (cit. on p. 64).

Loaiza-Ganem, Gabriel and John P. Cunningham (2019). "The continuous Bernoulli: fixing a pervasive error in variational autoencoders". In: *Advances in Neural Information Processing Systems*. Ed. by Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily Fox, and Roman Garnett. Vol. 32. Curran Associates, Inc., pp. 13287–13297 (cit. on p. 26).

Locatello, Francesco, Stefan Bauer, Mario Lucic, Gunnar Rätsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem (June 2019). "Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, United States of America: PMLR, pp. 4114–4124 (cit. on pp. 22, 23, 25).

Logeswaran, Lajanugen and Honglak Lee (2018). "An efficient framework for learning sentence representations". In: *International Conference on Learning Representations* (cit. on pp. 20, 41).

Long, Zichao, Yiping Lu, and Bi Dong (2019). "PDE-Net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network". In: *Journal of Computational Physics* 399 (cit. on pp. 17, 84).

Long, Zichao, Yiping Lu, Xianzhong Ma, and Bin Dong (July 2018). "PDE-Net: Learning PDEs from Data". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm, Sweden: PMLR, pp. 3208–3216 (cit. on pp. 17, 84, 207).

Lotter, William, Gabriel Kreiman, and David Cox (2017). "Deep Predictive Coding Networks for Video Prediction and Unsupervised Learning". In: *International Conference on Learning Representations* (cit. on p. 64).

Lu, Chaochao, Michael Hirsch, and Bernhard Schölkopf (July 2017). "Flexible Spatio-Temporal Networks for Video Prediction". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2137–2145 (cit. on p. 64).

Lu, Rui, Kailun Wu, Zhiyao Duan, and Changshui Zhang (Mar. 2017). "Deep ranking: Triplet MatchNet for music metric learning". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 121–125 (cit. on p. 40).

Lu, Yiping, Aoxiao Zhong, Quanzheng Li, and Bin Dong (July 2018). "Beyond Finite Layer Neural Networks: Bridging Deep Architectures and Numerical Differential Equations". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm, Sweden: PMLR, pp. 3276–3285 (cit. on p. 17).

Lucic, Mario, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet (2018). "Are GANs Created Equal? A Large-Scale Study". In: *Advances in Neural Information Processing Systems*. Ed. by Samy Bengio, Hanna Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett. Vol. 31. Curran Associates, Inc., pp. 698–707 (cit. on p. 103).

Lyu, Xinrui, Matthias Hüser, Stephanie L. Hyland, George Zerveas, and Gunnar Rätsch (2018). "Improving Clinical Predictions through Unsupervised Time Series Representation Learning". In: arXiv: 1812.00490 (cit. on p. 22).

Madec, Gurvan and NEMO System Team (2019). *NEMO ocean engine*. Tech. rep. 27. Scientific Notes of Climate Modelling Center, Institut Pierre-Simon Laplace (IPSL). Zenodo (cit. on p. 93).

Madsen, Andreas (2019). "Visualizing memorization in RNNs". In: *Distill* (cit. on p. 13).

Malhotra, Pankaj, Vishnu TV, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff (Apr. 2017). "TimeNet: Pre-trained deep recurrent neural network for time series classification". In: *25th European Symposium on Artificial Neural Networks — ESANN 2017*. Bruges, Belgium (cit. on pp. 22, 40, 41, 46, 47).

Mao, Xudong, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley (Oct. 2017). "Least Squares Generative Adversarial Networks". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 2813–2821 (cit. on p. 105).

Mathieu, Emile, Tom Rainforth, Narayanaswamy Siddharth, and Yee Whye Teh (June 2019). "Disentangling Disentanglement in Variational Autoencoders". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 4402–4412 (cit. on p. 25).

Mathieu, Michael, Camille Couprie, and Yann LeCun (2016). "Deep multi-scale video prediction beyond mean square error". In: *International Conference on Learning Representations* (cit. on pp. 6, 30, 64, 205).

Mescheder, Lars, Andreas Geiger, and Sebastian Nowozin (July 2018). "Which Training Methods for GANs do actually Converge?" In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 3481–3490 (cit. on pp. 104, 125).

Mescheder, Lars, Sebastian Nowozin, and Andreas Geiger (2017). "The Numerics of GANs". In: *Advances in Neural Information Processing Systems*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna Wallach, Rob Fergus, Swaminathan Venkata Narayana Vishwanathan, and Roman Garnett. Vol. 30. Curran Associates, Inc., pp. 1823–1833 (cit. on pp. 18, 104).

Metz, Luke, Ben Poole, David Pfau, and Jascha Sohl-Dickstein (2017). "Unrolled Generative Adversarial Networks". In: *International Conference on Learning Representations* (cit. on p. 237).

Micikevicius, Paulius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu (2018). "Mixed Precision Training". In: *International Conference on Learning Representations* (cit. on p. 71).

Mikolov, Tomáš, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur (2010). "Recurrent neural network based language model". In: *11th Annual Conference of the International Speech Communication Association 2010 (INTERSPEECH 2010)*. ISCA, pp. 1045–1048 (cit. on p. 11).

Mikolov, Tomáš, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeffrey Dean (2013). "Distributed Representations of Words and Phrases and their Compositionality". In: *Advances in Neural Information Processing Systems*. Ed. by Christopher J. C. Burges, Léon Bottou, Max Welling, Zoubin Ghahramani, and Kilian Q. Weinberger. Vol. 26. Curran Associates, Inc., pp. 3111–3119 (cit. on pp. 6, 20, 40, 41, 139).

Miller Jr., Willard (1983). "The Technique of Variable Separation for Partial Differential Equations". In: *Nonlinear Phenomena*. Ed. by Kurt Bernardo Wolf. Berlin - Heidelberg, Germany: Springer Berlin Heidelberg, pp. 184–208 (cit. on p. 85).

– (1988). "Mechanisms for Variable Separation in Partial Differential Equations and Their Relationship to Group Theory". In: *Symmetries and Nonlinear Phenomena: Proceedings of the International School on Applied Mathematics*. Ed. by Decio Levi and Pavel Winternitz. Singapore: World Scientific, pp. 188–221 (cit. on pp. 83, 85).

Minderer, Matthias, Chen Sun, Ruben Villegas, Forrester Cole, Kevin Murphy, and Honglak Lee (2019). "Unsupervised Learning of Object Structure and Dynamics from Videos". In: *Advances in Neural Information Processing Systems*. Ed. by Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily Fox, and Roman Garnett. Vol. 32. Curran Associates, Inc., pp. 92–102 (cit. on pp. 23, 65, 72, 77, 78, 82, 168, 169).

Mirza, Mehdi and Simon Osindero (2014). "Conditional Generative Adversarial Nets". In: arXiv: 1411.1784 (cit. on p. 30).

Mroueh, Youssef and Truyen Nguyen (Apr. 2021). "On the Convergence of Gradient Descent in GANs: MMD GAN As a Gradient Flow". In: *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*. Ed. by Arindam Banerjee and Kenji Fukumizu. Vol. 130. Proceedings of Machine Learning Research. PMLR, pp. 1720–1728 (cit. on p. 113).

Mroueh, Youssef, Tom Sercu, and Anant Raj (Apr. 2019). "Sobolev Descent". In: *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*. Ed. by Kamalika Chaudhuri and Masashi Sugiyama. Vol. 89. Proceedings of Machine Learning Research. PMLR, pp. 2976–2985 (cit. on pp. 116, 226).

Mrowca, Damian, Chengxu Zhuang, Elias Wang, Nick Haber, Li F Fei-Fei, Joshua B. Tenenbaum, and Daniel L. K. Yamins (2018). "Flexible Neural Representation for Physics Prediction". In: *Advances in Neural Information Processing Systems*. Ed. by Samy Bengio, Hanna Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett. Vol. 31. Curran Associates, Inc., pp. 8813–8824 (cit. on p. 22).

Muandet, Krikamol, Kenji Fukumizu, Bharath Sriperumbudur, and Bernhard Schölkopf (2017). "Kernel Mean Embedding of Distributions: A Review and Beyond". In: *Foundations and Trends® in Machine Learning* 10.1–2, pp. 1–141 (cit. on pp. 112, 119, 139).

Müller, Alfred (1997). "Integral Probability Metrics and Their Generating Classes of Functions". In: *Advances in Applied Probability* 29.2, pp. 429–443 (cit. on pp. 28, 105).

Nagabandi, Anusha, Ignasi Clavera, Simin Liu, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn (2019). "Learning to Adapt in Dynamic, Real-World Environments through Meta-Reinforcement Learning". In: *International Conference on Learning Representations* (cit. on p. 7).

Nagarajan, Vaishnavh and J. Zico Kolter (2017). "Gradient descent GAN optimization is locally stable". In: *Advances in Neural Information Processing Systems*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna Wallach, Rob Fergus, Swaminathan Venkata Narayana Vishwanathan, and Roman Garnett. Vol. 30. Curran Associates, Inc., pp. 5591–5600 (cit. on pp. 18, 104).

Nambiar, Ananthan, Maeve Heflin, Simon Liu, Sergei Maslov, Mark Hopkins, and Anna Ritz (2020). "Transforming the Language of Life: Transformer Neural Networks for Protein Prediction Tasks". In: *Proceedings of the 11th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*. BCB'20. Virtual event: Association for Computing Machinery (cit. on p. 20).

Norcliffe, Alexander, Cristian Bodnar, Ben Day, Jacob Moss, and Pietro Liò (2021). "Neural ODE Processes". In: *International Conference on Learning Representations* (cit. on p. 22).

Novak, Roman, Lechao Xiao, Jiri Hron, Jaehoon Lee, Alexander A. Alemi, Jascha Sohl-Dickstein, and Samuel S. Schoenholz (2020). "Neural Tangents: Fast and Easy Infinite Neural Networks in Python". In: *International Conference on Learning Representations*. URL: https://github.com/google/neural-tangents (cit. on pp. 104, 114, 138, 236, 237, 241).

Nowozin, Sebastian, Botond Cseke, and Ryota Tomioka (2016). "f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization". In: *Advances in Neural Information Processing Systems*. Ed. by Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett. Vol. 29. Curran Associates, Inc., pp. 271–279 (cit. on pp. 104, 105).

Odena, Augustus, Christopher Olah, and Jonathon Shlens (Aug. 2017). "Conditional Image Synthesis with Auxiliary Classifier GANs". 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. PMLR, pp. 2642–2651 (cit. on p. 30).

Oord, Aäron Van, Nal Kalchbrenner, and Koray Kavukcuoglu (June 2016). "Pixel Recurrent Neural Networks". In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria-Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, United States of America: PMLR, pp. 1747–1756 (cit. on p. 28).

Ott, Myle, Sergey Edunov, David Grangier, and Michael Auli (Oct. 2018). "Scaling Neural Machine Translation". In: *Proceedings of the Third Conference on Machine Translation: Research Papers*. Brussels, Belgium: Association for Computational Linguistics, pp. 1–9 (cit. on p. 19).

Pajot, Arthur (Dec. 2019). "Incorporating Physical Knowledge Into Deep Neural Network". PhD thesis. Sorbonne Université. URL: https://tel.archives-ouvertes.fr/tel-03015931 (cit. on p. 7).

Pashevich, Alexander, Cordelia Schmid, and Chen Sun (Oct. 2021). "Episodic Transformer for Vision-and-Language Navigation". In: *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15942–15952 (cit. on p. 140).

Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems*. Ed. by Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily Fox, and Roman Garnett. Vol. 32. Curran Associates, Inc., pp. 8026–8037 (cit. on pp. 5, 46, 71, 91).

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12.85, pp. 2825–2830 (cit. on p. 46).

Pineau, Joëlle, Philippe Vincent-Lamarre, Koustuv Sinha, Vincent Larivière, Alina Beygelzimer, Florence d'Alché-Buc, Emily Fox, and Hugo Larochelle (2021). "Improving Reproducibility in Machine Learning Research (A Report from the NeurIPS 2019 Reproducibility Program)". In: *Journal of Machine Learning Research* 22.164, pp. 1–20 (cit. on p. 132).

Pineda, Fernando J. (1988). "Generalization of Back propagation to Recurrent and Higher Order Neural Networks". In: *Neural Information Processing Systems*. Ed. by Dana Z. Anderson. American Institute of Physics, pp. 602–611 (cit. on p. 16).

Plessix, René-Edouard (Nov. 2006). "A review of the adjoint-state method for computing the gradient of a functional with geophysical applications". In: *Geophysical Journal International* 167.2, pp. 495–503 (cit. on p. 17).

Polyanin, Andrei D. (July 2019). "Functional separable solutions of nonlinear convection-diffusion equations with variable coefficients". In: *Communications in Nonlinear Science and Numerical Simulation* 73, pp. 379–390 (cit. on p. 85).

Polyanin, Andrei D. (2020). "Functional Separation of Variables in Nonlinear PDEs: General Approach, New Solutions of Diffusion-Type Equations". In: *Mathematics* 8.1 (cit. on p. 85).

Polyanin, Andrei D. and Valentin F. Zaitsev (2002). *Handbook of Exact Solutions for Ordinary Differential Equations*. 2nd edition. Chapman & Hall/CRC (cit. on p. 15).

Polyanin, Andrei D. and Alexei I. Zhurov (2020). "Separation of variables in PDEs using nonlinear transformations: Applications to reaction-diffusion type equations". In: *Applied Mathematics Letters* 100 (cit. on p. 85).

Pontryagin, Lev Semyonovich, Vladimir Grigorevich Boltyanskii, Revaz Valerianovic Gamkrelidze, and Evgenii Frolovich Mishechenko (1962). *Mathematical Theory of Optimal Processes*. English edition. International Series of Monographs in Pure and Applied Mathematics. John Wiley & Sons (cit. on p. 16).

Qin, Chongli, Yan Wu, Jost Tobias Springenberg, Andy Brock, Jeff Donahue, Timothy Lillicrap, and Pushmeet Kohli (2020). "Training Generative Adversarial Networks by Solving Ordinary Differential Equations". In: *Advances in Neural Information Processing Systems*. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. Vol. 33. Curran Associates, Inc., pp. 5599–5609 (cit. on p. 18).

Radford, Alec, Luke Metz, and Soumith Chintala (2016). "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: *International Conference on Learning Representations* (cit. on pp. 138, 139, 169, 202).

Raissi, Maziar (2018). "Deep Hidden Physics Models: Deep Learning of Nonlinear Partial Differential Equations". In: *Journal of Machine Learning Research* 19.25, pp. 1–24 (cit. on pp. 87, 196).

Raissi, Maziar, Paris Perdikaris, and George Em Karniadakis (2019). "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational Physics* 378, pp. 686–707 (cit. on p. 17).

Raissi, Maziar, Alireza Yazdani, and George Em Karniadakis (2020). "Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations". In: *Science* 367.6481, pp. 1026–1030 (cit. on p. 84).

Ranzato, Marc'Aurelio, Arthur Szlam, Joan Bruna, Michael Mathieu, Ronan Collobert, and Sumit Chopra (2014). "Video (Language) Modeling: a Baseline for Generative Models of Natural Videos". In: arXiv: 1412.6604 (cit. on p. 64).

Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra (June 2014). "Stochastic Backpropagation and Approximate Inference in Deep Generative Models". In: *Proceedings of the 31st International Conference on Machine Learning*. Ed. by Eric P. Xing and Tony Jebara. Vol. 32. Proceedings of Machine Learning Research. Beijing, China: PMLR, pp. 1278–1286 (cit. on p. 25).

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention — MICCAI 2015*. Ed. by Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi. Cham, Switzerland: Springer International Publishing, pp. 234–241 (cit. on pp. 69, 202).

Rousseau, François, Lucas Drumetz, and Ronan Fablet (2019). "Residual Networks as Flows of Diffeomorphisms". In: *Journal of Mathematical Imaging and Vision* 62, pp. 365–375 (cit. on p. 70).

Rubanova, Yulia, Ricky T. Q. Chen, and David Duvenaud (2019). "Latent Ordinary Differential Equations for Irregularly-Sampled Time Series". In: *Advances in Neural Information Processing Systems*. Ed. by Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily Fox, and Roman Garnett. Vol. 32. Curran Associates, Inc., pp. 5320–5330 (cit. on pp. 63, 66, 90).

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams (1986). "Learning Representations by Back-Propagating Errors". In: *Nature* 323, pp. 533–536 (cit. on pp. 11, 16).

Rusch, T. Konstantin and Siddhartha Mishra (July 2021). "UnICORNN: A recurrent model for learning *very* long time dependencies". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, pp. 9168–9178 (cit. on p. 17).

Russakovsky, Olga, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei (2015). "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision* 115, pp. 211–252 (cit. on p. 5).

Russell, Stuart and Peter Norvig (2020). *Artificial Intelligence: A Modern Approach*. 4th edition. Pearson Series in Artificial Intelligence. Pearson (cit. on p. 5).

Ruthotto, Lars and Eldad Haber (2020). "Deep Neural Networks Motivated by Partial Differential Equations". In: *Journal of Mathematical Imaging and Vision* 62, pp. 352–364 (cit. on p. 17).

Ryder, Tom, Andrew Golightly, A. Stephen McGough, and Dennis Prangle (July 2018). "Black-Box Variational Inference for Stochastic Differential Equations". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm, Sweden: PMLR, pp. 4423–4432 (cit. on pp. 18, 66).

Saha, Priyabrata, Saurabh Dash, and Saibal Mukhopadhyay (2020). "PhICNet: Physics-Incorporated Convolutional Recurrent Neural Networks for Modeling Dynamical Systems". In: arXiv: 2004.06243 (cit. on pp. 93, 198).

Salakhutdinov, Ruslan and Geoffrey E. Hinton (Apr. 2009). "Deep Boltzmann Machines". In: *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*. Ed. by David van Dyk and Max Welling. Vol. 5. Proceedings of Machine Learning Research. Hilton Clearwater Beach Resort, Clearwater Beach, Florida, United States of America: PMLR, pp. 448–455 (cit. on p. 28).

Salimans, Tim and Diederik P. Kingma (2016). "Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett. Vol. 29. Curran Associates, Inc., pp. 901–909 (cit. on p. 44).

Sander, Michael E., Pierre Ablin, Mathieu Blondel, and Gabriel Peyré (July 2021). "Momentum Residual Neural Networks". In: *Proceedings of the 38th International*

Conference on Machine Learning. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, pp. 9276–9287 (cit. on p. 17).

Santoro, Adam, David Raposo, David G. T. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap (2017). "A simple neural network module for relational reasoning". In: *Advances in Neural Information Processing Systems*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna Wallach, Rob Fergus, Swaminathan Venkata Narayana Vishwanathan, and Roman Garnett. Vol. 30. Curran Associates, Inc., pp. 4974–4983 (cit. on p. 69).

Saunshi, Nikunj, Orestis Plevrakis, Sanjeev Arora, Mikhail Khodak, and Hrishikesh Khandeparkar (June 2019). "A Theoretical Analysis of Contrastive Unsupervised Representation Learning". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 5628–5637 (cit. on p. 40).

Schäfer, Patrick (2015). "The BOSS is concerned with time series classification in the presence of noise". In: *Data Mining and Knowledge Discovery* 29, pp. 1505–1530 (cit. on p. 47).

Scheuerer, Michael (Oct. 2009). "A Comparison of Models and Methods for Spatial Interpolation in Statistics and Numerical Analysis". PhD thesis. Georg-August-Universität Göttingen. URL: https://ediss.uni-goettingen.de/handle/11858/00-1735-0000-0006-B3D5-1 (cit. on p. 219).

Schmidhuber, Jürgen (2015). "Deep learning in neural networks: An overview". In: *Neural Networks* 61, pp. 85–117 (cit. on p. 6).

Schroff, Florian, Dmitry Kalenichenko, and James Philbin (June 2015). "FaceNet: A Unified Embedding for Face Recognition and Clustering". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 815–823 (cit. on pp. 40, 41).

Schüldt, Christian, Ivan Laptev, and Barbara Caputo (Aug. 2004). "Recognizing Human Actions: A Local SVM Approach". In: *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.* Vol. 3, pp. 32–36 (cit. on pp. 74, 168).

Schuster, Mike (2000). "Better Generative Models for Sequential Data Problems: Bidirectional Recurrent Mixture Density Networks". In: *Advances in Neural Information Processing Systems*. Ed. by Sara A. Solla, Todd K. Leen, and Klaus-Robert Müller. Vol. 12. MIT Press, pp. 589–594 (cit. on p. 31).

Schuster, Mike and Kuldip K. Paliwal (1997). "Bidirectional Recurrent Neural Networks". In: *IEEE Transactions on Signal Processing* 45.11, pp. 2673–2681 (cit. on p. 14).

Seo, Youngjoo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson (2018). "Structured Sequence Modeling with Graph Convolutional Recurrent Networks". In: *Neural Information Processing*. Ed. by Long Cheng, Andrew Chi Sing Leung, and Seiichi Ozawa. Cham, Switzerland: Springer International Publishing, pp. 362–373 (cit. on p. 14).

Sermanet, Pierre, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, and Sergey Levine (May 2018). "Time-Contrastive Networks: Self-Supervised Learning from Video". In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1134–1141 (cit. on p. 21).

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*. Ed. by Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett. Vol. 28. Curran Associates, Inc., pp. 802–810 (cit. on pp. 14, 17, 64, 95, 98).

Siegelmann, Hava T. and Eduardo D. Sontag (1994). "Analog computation via neural networks". In: *Theoretical Computer Science* 131.2, pp. 331–360 (cit. on p. 12).

Silver, David, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis (2016). "Mastering the game of Go with deep neural networks and tree search". In: *Nature* 529, pp. 484–489 (cit. on p. 5).

Silver, David, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis (2017). "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm". In: arXiv: 1712.01815 (cit. on p. 5).

Simonyan, Karen and Andrew Zisserman (2015). "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *International Conference on Learning Representations* (cit. on pp. 169, 202).

Sirignano, Justin and Konstantinos Spiliopoulos (2018). "DGM: A deep learning algorithm for solving partial differential equations". In: *Journal of Computational Physics* 375, pp. 1339–1364 (cit. on pp. 86, 87, 196).

Sitzmann, Vincent, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein (2020). "Implicit Neural Representations with Periodic Activation Functions". In: *Advances in Neural Information Processing Systems*. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. Vol. 33. Curran Associates, Inc., pp. 7462–7473 (cit. on p. 139).

Sohl-Dickstein, Jascha, Roman Novak, Samuel S. Schoenholz, and Jaehoon Lee (2020). "On the infinite width limit of neural networks with a standard parameterization". In: arXiv: 2001.07301 (cit. on p. 104).

Sohn, Kihyuk, Honglak Lee, and Xinchen Yan (2015). "Learning Structured Output Representation using Deep Conditional Generative Models". In: *Advances in Neural Information Processing Systems*. Ed. by Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett. Vol. 28. Curran Associates, Inc., pp. 3483–3491 (cit. on p. 30).

Sønderby, Casper Kaae, Jose Caballero, Lucas Theis, Wenzhe Shi, and Ferenc Huszár (2017). "Amortised MAP Inference for Image Super-resolution". In: *International Conference on Learning Representations* (cit. on pp. 112, 238).

Sønderby, Casper Kaae, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther (2016). "Ladder Variational Autoencoders". In: *Advances in Neural Information Processing Systems*. Ed. by Daniel D. Lee, Masashi Sugiyama, Ulrike von

Luxburg, Isabelle Guyon, and Roman Garnett. Vol. 29. Curran Associates, Inc., pp. 3745–3753 (cit. on p. 32).

Song, Yang, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole (2021). "Score-Based Generative Modeling through Stochastic Differential Equations". In: *International Conference on Learning Representations* (cit. on pp. 28, 141).

Sriperumbudur, Bharath K., Kenji Fukumizu, and Gert R. G. Lanckriet (2011). "Universality, Characteristic Kernels and RKHS Embedding of Measures". In: *Journal of Machine Learning Research* 12.70, pp. 2389–2410 (cit. on pp. 239, 240).

Sriperumbudur, Bharath K., Arthur Gretton, Kenji Fukumizu, Bernhard Schölkopf, and Gert R. G. Lanckriet (2010). "Hilbert Space Embeddings and Metrics on Probability Measures". In: *Journal of Machine Learning Research* 11.50, pp. 1517–1561 (cit. on p. 108).

Srivastava, Nitish, Elman Mansimov, and Ruslan Salakhudinov (July 2015). "Unsupervised Learning of Video Representations using LSTMs". In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, pp. 843–852 (cit. on pp. 11, 22, 40, 64, 74).

Steinwart, Ingo (Nov. 2001). "On the Influence of the Kernel on the Consistency of Support Vector Machines". In: *Journal of Machine Learning Research* 2, pp. 67–93 (cit. on p. 239).

Su, Weijie, Stephen Boyd, and Emmanuel J. Candès (2016). "A Differential Equation for Modeling Nesterov's Accelerated Gradient Method: Theory and Insights". In: *Journal of Machine Learning Research* 17.153, pp. 1–43 (cit. on p. 18).

Sun, Deqing, Stefan Roth, John P. Lewis, and Michael J. Black (2008). "Learning Optical Flow". In: *The European Conference on Computer Vision (ECCV)*. Ed. by David Forsyth, Philip Torr, and Andrew Zisserman. Berlin - Heidelberg, Germany: Springer Berlin Heidelberg, pp. 83–97 (cit. on p. 17).

Sun, Ruoyu, Tiantian Fang, and Alexander Schwing (2020). "Towards a Better Global Loss Landscape of GANs". In: *Advances in Neural Information Processing Systems*. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. Vol. 33. Curran Associates, Inc., pp. 10186–10198 (cit. on p. 104).

Szegedy, Christian, Sergey Ioffe, Vincent Vanhoucke, and Alexander A. Alemi (2017). "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning". In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. AAAI'17. San Francisco, California, United States of America: AAAI Press, pp. 4278–4284 (cit. on p. 5).

Tallec, Corentin and Yann Ollivier (2018). "Can recurrent neural networks warp time?" In: *International Conference on Learning Representations* (cit. on p. 17).

Tan, Mingxing and Quoc V. Le (June 2019). "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 6105–6114 (cit. on p. 19).

Tancik, Matthew, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng (2020). "Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains". In: *Advances in Neural Information Processing Systems*. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. Vol. 33. Curran Associates, Inc., pp. 7537–7547 (cit. on pp. 105, 139).

Tomczak, Jakub M. and Max Welling (Apr. 2018). "VAE with a VampPrior". In: *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*. Ed. by Amos Storkey and Fernando Perez-Cruz. Vol. 84. Proceedings of Machine Learning Research. PMLR, pp. 1214–1223 (cit. on p. 26).

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 (cit. on p. 84).

Toth, Peter, Danilo Jimenez Rezende, Andrew Jaegle, Sébastien Racanière, Aleksandar Botev, and Irina Higgins (2020). "Hamiltonian Generative Networks". In: *International Conference on Learning Representations* (cit. on p. 84).

Tulyakov, Sergey, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz (June 2018). "MoCoGAN: Decomposing Motion and Content for Video Generation". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1526–1535 (cit. on pp. 23, 64).

Turpault, Nicolas, Romain Serizel, and Emmanuel Vincent (May 2019). "Semi-supervised Triplet Loss Based Learning of Ambient Audio Embeddings". In: *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 760–764 (cit. on p. 40).

Unterthiner, Thomas, Sjoerd van Steenkiste, Karol Kurach, Raphaël Marinier, Marcin Michalski, and Sylvain Gelly (2018). "Towards Accurate Generative Models of Video: A New Metric & Challenges". In: arXiv: 1812.01717 (cit. on p. 72).

Vahdat, Arash and Jan Kautz (2020). "NVAE: A Deep Hierarchical Variational Autoencoder". In: *Advances in Neural Information Processing Systems*. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. Vol. 33. Curran Associates, Inc., pp. 19667–19679 (cit. on p. 26).

Oord, Aäron van den, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu (2016). "WaveNet: A Generative Model for Raw Audio". In: arXiv: 1609.03499 (cit. on pp. 19, 30, 39, 43, 44).

Oord, Aäron van den, Nal Kalchbrenner, Lasse Espeholt, Koray Kavukcuoglu, Oriol Vinyals, and Alex Graves (2016). "Conditional Image Generation with PixelCNN Decoders". In: *Advances in Neural Information Processing Systems*. Ed. by Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett. Vol. 29. Curran Associates, Inc., pp. 4797–4805 (cit. on pp. 28, 65).

Oord, Aäron van den, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis C. Cobo, Florian Stimberg, Norman Casagrande, Dominik Grewe, Seb Noury, Sander Dieleman, Erich Elsen, Nal Kalchbrenner, Heiga Zen, Alex Graves, Helen King, Tom Walters, Dan Belov, and Demis Hassabis (July 2018). "Parallel WaveNet: Fast High-Fidelity Speech Synthesis". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 3918–3926 (cit. on pp. 6, 19).

Oord, Aäron van den, Yazhe Li, and Oriol Vinyals (2018). "Representation Learning with Contrastive Predictive Coding". In: arXiv: 1807.03748 (cit. on pp. 21, 40).

Maaten, Laurens van der and Geoffrey E. Hinton (2008). "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9.86, pp. 2579–2605 (cit. on p. 52).

Van Gansbeke, Wouter, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool (Aug. 2020). "SCAN: Learning to Classify Images Without Labels". In: *The European Conference on Computer Vision (ECCV)*. Ed. by Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm. Cham, Switzerland: Springer International Publishing, pp. 268–285 (cit. on p. 6).

Steenkiste, Sjoerd van, Michael Chang, Klaus Greff, and Jürgen Schmidhuber (2018). "Relational Neural Expectation Maximization: Unsupervised Discovery of Objects and their Interactions". In: *International Conference on Learning Representations* (cit. on pp. 23, 64).

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna Wallach, Rob Fergus, Swaminathan Venkata Narayana Vishwanathan, and Roman Garnett. Vol. 30. Curran Associates, Inc., pp. 6000–6010 (cit. on pp. 6, 19, 28).

Villani, Cédric (2009). "The Wasserstein distances". In: *Optimal Transport: Old and New*. Grundlehren der mathematischen Wissenschaften. Berlin - Heidelberg, Germany: Springer Berlin Heidelberg, pp. 93–111 (cit. on p. 113).

Villegas, Ruben, Arkanath Pathak, Harini Kannan, Dumitru Erhan, Quoc V. Le, and Honglak Lee (2019). "High Fidelity Video Prediction with Large Stochastic Recurrent Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily Fox, and Roman Garnett. Vol. 32. Curran Associates, Inc., pp. 81–91 (cit. on pp. 65, 78, 140).

Villegas, Ruben, Jimei Yang, Seunghoon Hong, Xunyu Lin, and Honglak Lee (2017). "Decomposing Motion and Content for Natural Video Sequence Prediction". In: *International Conference on Learning Representations* (cit. on pp. 22, 23, 40, 64).

Villegas, Ruben, Jimei Yang, Yuliang Zou, Sungryull Sohn, Xunyu Lin, and Honglak Lee (Aug. 2017). "Learning to Generate Long-term Future via Hierarchical Prediction". 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. 3560–3569 (cit. on p. 23).

Voelker, Aaron R., Ivana Kajić, and Chris Eliasmith (2019). "Legendre Memory Units: Continuous-Time Representation in Recurrent Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily Fox, and Roman Garnett. Vol. 32. Curran Associates, Inc., pp. 15570–15579 (cit. on p. 17).

Vondrick, Carl, Hamed Pirsiavash, and Antonio Torralba (2016). "Generating Videos with Scene Dynamics". In: *Advances in Neural Information Processing Systems*. Ed. by Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett. Vol. 29. Curran Associates, Inc., pp. 613–621 (cit. on p. 23).

Vondrick, Carl and Antonio Torralba (July 2017). "Generating the Future with Adversarial Transformers". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2992–3000 (cit. on pp. 30, 64).

Waibel, Alexander, Toshiyuki Hanazawa, Geoffrey E. Hinton, Kiyohiro Shikano, and Kevin J. Lang (1989). "Phoneme Recognition Using Time-Delay Neural Networks". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 37.3, pp. 328–339 (cit. on p. 19).

Walker, Jacob, Carl Doersch, Abhinav Gupta, and Martial Hebert (Oct. 2016). "An Uncertain Future: Forecasting from Static Images Using Variational Autoencoders". In: *The European Conference on Computer Vision (ECCV)*. Ed. by Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling. Springer International Publishing, pp. 835–851 (cit. on p. 64).

Walker, Jacob, Abhinav Gupta, and Martial Hebert (Dec. 2015). "Dense Optical Flow Prediction From a Static Image". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 2443–2451 (cit. on p. 64).

Wang, Chuang, Hong Hu, and Yue M. Lu (2019). "A Solvable High-Dimensional Model of GAN". In: *Advances in Neural Information Processing Systems*. Ed. by Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily Fox, and Roman Garnett. Vol. 32. Curran Associates, Inc., pp. 13787–13796 (cit. on p. 18).

Wang, Rui, Karthik Kashinath, Mustafa Mustafa, Adrian Albert, and Rose Yu (2020). "Towards Physics-Informed Deep Learning for Turbulent Flow Prediction". In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD'20. Virtual event: Association for Computing Machinery, pp. 1457–1466 (cit. on p. 17).

Wang, Ting-Chun, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro (2018). "Video-to-Video Synthesis". In: *Advances in Neural Information Processing Systems*. Ed. by Samy Bengio, Hanna Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett. Vol. 31. Curran Associates, Inc., pp. 1152–1164 (cit. on pp. 30, 64).

Wang, Tongzhou and Phillip Isola (July 2020). "Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 9929–9939 (cit. on pp. 20, 41).

Wang, Yunbo, Zhifeng Gao, Mingsheng Long, Jianmin Wang, and Philip S. Yu (July 2018). "PredRNN++: Towards A Resolution of the Deep-in-Time Dilemma in Spatiotemporal Predictive Learning". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm, Sweden: PMLR, pp. 5123–5132 (cit. on pp. 98, 201).

Wang, Yunbo, Lu Jiang, Ming-Hsuan Yang, Li-Jia Li, Mingsheng Long, and Li Fei-Fei (2019). "Eidetic 3D LSTM: A Model for Video Prediction and Beyond". In: *International Conference on Learning Representations* (cit. on p. 98).

Wang, Yunbo, Mingsheng Long, Jianmin Wang, Zhifeng Gao, and Philip S. Yu (2017). "PredRNN: Recurrent Neural Networks for Predictive Learning using Spatiotemporal LSTMs". In: *Advances in Neural Information Processing Systems*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna Wallach, Rob Fergus, Swaminathan Venkata Narayana Vishwanathan, and Roman Garnett. Vol. 30. Curran Associates, Inc., pp. 879–888 (cit. on p. 98).

Wang, Yunbo, Jianjin Zhang, Hongyu Zhu, Mingsheng Long, Jianmin Wang, and Philip S. Yu (June 2019). "Memory in Memory: A Predictive Neural Network for Learning Higher-Order Non-Stationarity From Spatiotemporal Dynamics". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9146–9154 (cit. on pp. 93, 98, 201).

Wang, Zhengwei, Qi She, and Tomás E. Ward (Apr. 2021). "Generative Adversarial Networks in Computer Vision: A Survey and Taxonomy". In: *ACM Computing Surveys* 54.2 (cit. on p. 27).

Wang, Zhiguang, Zhiguang Yan, and Tim Oates (May 2017). "Time series classification from scratch with deep neural networks: A strong baseline". In: *International Joint Conference on Neural Networks (IJCNN)*, pp. 1578–1585 (cit. on pp. 19, 46, 48).

Weissenborn, Dirk, Oscar Täckström, and Jakob Uszkoreit (2020). "Scaling Autoregressive Video Models". In: *International Conference on Learning Representations* (cit. on pp. 6, 20, 65, 140).

Wichers, Nevan, Ruben Villegas, Dumitru Erhan, and Honglak Lee (July 2018). "Hierarchical Long-term Video Prediction without Supervision". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm, Sweden: PMLR, pp. 6038–6046 (cit. on p. 64).

Wöhler, Christian and Joachim K. Anlauf (2001). "Real-time object recognition on image sequences with the adaptable time delay neural network algorithm — applications for autonomous vehicles". In: *Image and Vision Computing* 19.9, pp. 593–618 (cit. on p. 19).

Wu, Yi-Fu, Jaesik Yoon, and Sungjin Ahn (July 2021). "Generative Video Transformer: Can Objects be the Words?" In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, pp. 11307–11318 (cit. on p. 20).

Wu, Ledell Yu, Adam Fisch, Sumit Chopra, Keith Adams, Antoine Bordes, and Jason Weston (2018). "StarSpace: Embed All The Things!" In: *Proceedings of the Thirty-*

Second AAAI Conference on Artificial Intelligence. AAAI'18. New Orleans, Louisiana, United States of America: AAAI Press, pp. 5569–5577 (cit. on p. 40).

Wu, Lingfei, Ian En-Hsu Yen, Jinfeng Yi, Fangli Xu, Qi Lei, and Michael J. Witbrock (Apr. 2018). "Random Warping Series: A Random Features Method for Time-Series Embedding". In: *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*. Ed. by Amos Storkey and Fernando Perez-Cruz. Vol. 84. Proceedings of Machine Learning Research. PMLR, pp. 793–802 (cit. on pp. 40, 46, 47).

Wu, Neo, Bradley Green, Xue Ben, and Shawn O'Banion (2020). "Deep Transformer Models for Time Series Forecasting: The Influenza Prevalence Case". In: arXiv: 2001.08317 (cit. on p. 20).

Wu, Yonghui, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughe, and Jeffrey Dean (2016). "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation". In: arXiv: 1609.08144 (cit. on p. 5).

Wu, Yue, Rongrong Gao, Jaesik Park, and Qifeng Chen (June 2020). "Future Video Synthesis With Object Motion Prediction". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5538–5547 (cit. on p. 64).

Xie, You, Erik Franz, Mengyu Chu, and Nils Thuerey (July 2018). "TempoGAN: A Temporally Coherent, Volumetric GAN for Super-Resolution Fluid Flow". In: *ACM Transactions on Graphics* 37.4 (cit. on p. 17).

Xu, Jingwei, Bingbing Ni, Zefan Li, Shuo Cheng, and Xiaokang Yang (June 2018). "Structure Preserving Video Prediction". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1460–1469 (cit. on p. 64).

Xu, Jingwei, Bingbing Ni, and Xiaokang Yang (2018). "Video Prediction via Selective Sampling". In: *Advances in Neural Information Processing Systems*. Ed. by Samy Bengio, Hanna Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett. Vol. 31. Curran Associates, Inc., pp. 1712–1722 (cit. on p. 64).

Xu, Tianlin, Li Kevin Wenliang, Michael Munn, and Beatrice Acciaio (2020). "COT-GAN: Generating Sequential Data via Causal Optimal Transport". In: *Advances in Neural Information Processing Systems*. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. Vol. 33. Curran Associates, Inc., pp. 8798–8809 (cit. on p. 30).

Xu, Wei, Xin Liu, and Yihong Gong (2003). "Document Clustering Based on Non-negative Matrix Factorization". In: *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*. SIGIR'03. Toronto, Ontario, Canada: Association for Computing Machinery, pp. 267–273 (cit. on p. 46).

Xu, Zhenjia, Zhijian Liu, Chen Sun, Kevin Murphy, William T. Freeman, Joshua B. Tenenbaum, and Jiajun Wu (2019). "Unsupervised Discovery of Parts, Structure,

and Dynamics". In: *International Conference on Learning Representations* (cit. on pp. 23, 65).

Xue, Tianfan, Jiajun Wu, Katherine L. Bouman, and William T. Freeman (2016). "Visual Dynamics: Probabilistic Future Frame Synthesis via Cross Convolutional Networks". In: *Advances in Neural Information Processing Systems*. Ed. by Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett. Vol. 29. Curran Associates, Inc., pp. 91–99 (cit. on p. 65).

Yang, Greg (2020). "Tensor Programs II: Neural Tangent Kernel for Any Architecture". In: arXiv: 2006.14548 (cit. on pp. 104, 236).

Yang, Greg and Edward J. Hu (July 2021). "Tensor Programs IV: Feature Learning in Infinite-Width Neural Networks". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, pp. 11727–11737 (cit. on pp. 105, 236).

Yang, Greg and Hadi Salman (2019). "A Fine-Grained Spectral Perspective on Neural Networks". In: arXiv: 1907.10599 (cit. on pp. 105, 125, 239).

Yang, Zhilin, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ Salakhutdinov, and Quoc V. Le (2019). "XLNet: Generalized Autoregressive Pretraining for Language Understanding". In: *Advances in Neural Information Processing Systems*. Ed. by Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily Fox, and Roman Garnett. Vol. 32. Curran Associates, Inc., pp. 5753–5763 (cit. on p. 19).

Yin, Yuan, Arthur Pajot, Emmanuel de Bézenac, and Patrick Gallinari (2020). *Unsupervised Spatiotemporal Data Inpainting*. URL: https://openreview.net/forum?id=rylqmxBKvH (cit. on p. 19).

Yingzhen, Li and Stephan Mandt (July 2018). "Disentangled Sequential Autoencoder". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm, Sweden: PMLR, pp. 5670–5679 (cit. on pp. 22, 23, 69).

Yıldız, Çağatay, Markus Heinonen, and Harri Lähdesmäki (2019). "ODE^{2}VAE: Deep generative second order ODEs with Bayesian neural networks". In: *Advances in Neural Information Processing Systems*. Ed. by Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily Fox, and Roman Garnett. Vol. 32. Curran Associates, Inc., pp. 13434–13443 (cit. on pp. 66, 83).

Young, Tom, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria (2018). "Recent Trends in Deep Learning Based Natural Language Processing". In: *IEEE Computational Intelligence Magazine* 13.3, pp. 55–75 (cit. on p. 39).

Yu, Fisher and Vladlen Koltun (2016). "Multi-Scale Context Aggregation by Dilated Convolutions". In: *International Conference on Learning Representations* (cit. on pp. 19, 43).

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: *ECCV 2016 Workshops*. Ed. by Gang Hua and Hervé Jégou. Cham, Switzerland: Springer International Publishing, pp. 3–10 (cit. on p. 17).

Zaheer, Manzil, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J. Smola (2017). "Deep Sets". In: *Advances in Neural Information Processing Systems*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna Wallach, Rob Fergus, Swaminathan Venkata Narayana Vishwanathan, and Roman Garnett. Vol. 30. Curran Associates, Inc., pp. 3394–3404 (cit. on p. 69).

Zhang, Junbo, Yu Zheng, and Dekang Qi (2017). "Deep Spatio-Temporal Residual Networks for Citywide Crowd Flows Prediction". In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. AAAI'17. San Francisco, California, United States of America: AAAI Press, pp. 1655–1661 (cit. on pp. 97, 201).

Zhang, Richard, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang (June 2018). "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 586–595 (cit. on p. 72).

Zhang, Xiang, Junbo Zhao, and Yann LeCun (2015). "Character-level Convolutional Networks for Text Classification". In: *Advances in Neural Information Processing Systems*. Ed. by Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett. Vol. 28. Curran Associates, Inc., pp. 649–657 (cit. on p. 19).

Zhang, Yaoyu, Zhi-Qin John Xu, Tao Luo, and Zheng Ma (July 2020). "A type of generalization error induced by initialization in deep neural networks". In: *Proceedings of The First Mathematical and Scientific Machine Learning Conference*. Ed. by Jianfeng Lu and Rachel Ward. Vol. 107. Proceedings of Machine Learning Research. Princeton University, Princeton, New Jersey, United States of America: PMLR, pp. 144–164 (cit. on pp. 112, 116, 239, 242).

Zhou, Yufan, Zhenyi Wang, Jiayi Xian, Changyou Chen, and Jinhui Xu (2021). "Meta-Learning with Neural Tangent Kernels". In: *International Conference on Learning Representations* (cit. on p. 139).

Zhou, Zhiming, Jiadong Liang, Yuxuan Song, Lantao Yu, Hongwei Wang, Weinan Zhang, Yong Yu, and Zhihua Zhang (June 2019). "Lipschitz Generative Adversarial Nets". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, United States of America: PMLR, pp. 7584–7593 (cit. on p. 104).

Zhu, Jun-Yan, Taesung Park, Phillip Isola, and Alexei A. Efros (Oct. 2017). "Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 2242–2251 (cit. on p. 6).

Zhuang, Juntang, Nicha Dvornek, Xiaoxiao Li, Sekhar Tatikonda, Xenophon Papademetris, and James Duncan (July 2020). "Adaptive Checkpoint Adjoint Method for Gradient Estimation in Neural ODE". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 11639–11649 (cit. on p. 17).